Ingeniería Informática

PROYECTO FIN DE CARRERA

APLICACIÓN DEL ESTÁNDAR VOICEXML PARA EL DESARROLLO DE UN SISTEMA DE ADMINISTRACIÓN DE FINCAS

Autor: Felipe Carlos Ballesteros Costero

Tutor: Dr. David Griol Barres

Septiembre 2017

Título: Aplicación del estándar VoiceXML para el desarrollo de un sistema de administración de fincas.
Autor: Felipe Carlos Ballesteros Costero.
Director: Dr. David Griol Barres.

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de septiembre de 2017 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Quiero agradecer en primer lugar al tutor de este proyecto, David, por darme la oportunidad de involucrarme en este trabajo y, fundamentalmente, por su paciencia y apoyo durante todo este tiempo.

También agradecer a mi compañero, Alberto, cuya compañía y ayuda durante los años en la universidad han sido fundamentales para poder haber llegado hasta aquí.

A mí mujer, Laura, que ha logrado que el día a día sea mucho más fácil y cuya confianza y cariño sean de gran ayuda para solventar todo lo que haya que afrontar.

Finalmente, quiero agradecer especialmente a mis padres, Alejandro y Amelia, y a mis hermanas, Sara y Gemma, porque su apoyo, su cariño, y su insistencia y ánimo, han logrado que por fin llegue este momento.

A todos ellos, GRACIAS.

Resumen

Este Proyecto Final de Carrera nace con el objetivo de facilitar y mejorar la administración de fincas mediante un lenguaje de acceso vocal como es VoiceXML.

Existen numerosas empresas dedicadas a la administración de fincas y muchas de ellas se ayudan de programas de gestión que ayudan en la planificación y optimización de los recursos físicos, técnicos y humanos. El sistema desarrollado en este proyecto ofrece una nueva funcionalidad gracias a la cual se facilitan estos servicios tanto al administrador como a los usuarios a través de un teléfono mediante la voz.

Este sistema de comunicación multimodal permite diferenciar entre distintos usuarios ofreciendo el acceso a la información en tiempo real sobre la comunidad, las propiedades, los distintos contratos y proveedores, la junta de gobierno, las juntas vecinales y la información contable.

La aplicación desarrollada, además del estándar VoiceXML, utiliza otras tecnologías como las bases de datos (MySQL), servidores web y de VoiceXML (000webhost y Voxeo Evolution) y uso de diferentes lenguajes de programación como SQL y PHP.

El proyecto se complementa con un estudio detallado de los sistemas de diálogo y de la aplicación del estándar VoiceXML para entender mejor los desarrollos llevados a cabo en este Proyecto Final de Carrera.

Palabras clave: acceso vocal, VoiceXML, comunicación multimodal, administración y gestión de fincas, usuarios, bases de datos, lenguajes de programación, sistemas de diálogo.

Abstract

This Final Degree Project begins with de goal of facilitating and improving property management through a voice access lenguaje such as VoiceXML.

There are a lot of companies dedicated to property management and many of them are aided by management programs that facilitate the planning and optimization of physical, technical and human resources. The system developed in this project offers a new functionality that allows users and administrators to use these services through a telephone by voice.

This multimodal communication system allows to differentiate between different users offering access to real-time information about the community, properties, contracts and suppliers, the government board, neighborhood meetings and accounting information.

The application developed, in addition to VoiceXML standard, uses another technologies such as databases (MySQL), web servers and VoiceXML (000webhost and Voxeo Evolution) and use different programming languages such as SQL and PHP.

The project is complemented by a detailed study of dialogue systems and the application of the VoiceXML standard to better understand the developments of this Final Degree Project.

Keywords: voice access, VoiceXML, multimodal communication, property management, users, databases, programming languages, dialogue systems.

Índice general

1. INTRODUCCIÓN Y OBJETIVOS 15
1.1 Introducción 15
1.2 Objetivos 17
1.3 Fases del desarrollo 18
1.4 Planificación temporal 20
1.5 Medios empleados 21
1.6 Estructura de la memoria 22
2. ESTADO DEL ARTE 24
2.1 Sistemas de Diálogo 24
2.1.1 Introducción 24
2.1.2 Arquitectura de un sistema de diálogo 25
2.1.3 Aplicaciones de los sistemas de diálogo 28
2.2 Voice Extensible Markup Language (VoiceXML) 31
2.2.1 Introducción al estándar VoiceXML 31
2.2.2 Ventajas e inconvenientes de VoiceXML 34
2.2.3 Aplicaciones adecuadas de VoiceXML 35
2.2.4 Conceptos básicos de VoiceXML 36
2.2.5 Elementos de VoiceXML 39
2.3 Plataforma Voxeo Evolution 42
2.3.1 Introducción 42
2.3.2 Funcionamiento de Voxeo Evolution 43
2.3.3 Desarrollo de una aplicación VoiceXML en Voxeo 43
3. DESCRIPCIÓN GENERAL DEL SISTEMA DESARROLLADO 48
3.1 Funcionalidad y arquitectura 48
3.2 Servidor Web 50
3.3 Base de datos 51
4. EXPLICACIÓN DETALLADA DEL SISTEMA DESARROLLADO 59
4.1 Descripción general 59
4.2 Módulos del sistema 61
4.2.1 Login 61

4.2.2 Comunidad *64*
4.2.3 Propiedades *66*
4.2.4 Cargos *70*
4.2.5 Proveedores *73*
4.2.6 Contratos *79*
4.2.7 Juntas Vecinales *83*
4.2.8 Contabilidad *89*
5. EVALUACIÓN DEL SISTEMA 95
5.1 Metodología de la evaluación 95
5.2 Resultados de la evaluación 97
6. CONCLUSIONES Y FUTUROS PROYECTOS 99
6.1 Conclusiones 99
6.2 Trabajos futuros 101
7. PRESUPUESTO 103
8. GLOSARIO 106
9. REFERENCIAS 107

Índice de figuras

Figura 1. Ejemplos de programas de administración de fincas que hay actualmente en el mercado. 17
Figura 2. Ejemplo de programa de administración de fincas que hay actualmente en el mercado. 18
Figura 3. Arquitectura de un sistema de diálogo 26
Figura 4. Esquema de un sistema de reconocimiento del habla 26
Figura 5. Esquema del procesamiento lingüístico en un sintetizador de texto en habla [LLIS03] 28
Figura 6. Algunos clientes y aplicaciones de la empresa Natural Vox. 29
Figura 7. Arquitectura de un sistema VoiceXML 32
Figura 8. Ejemplo de un programa de VoiceXML 33
Figura 9. Interpretación de una página web 43
Figura 10. Interpretación de una página VoiceXML 43
Figura 11. Acceso a la cuenta de Voxeo 44
Figura 12. Página principal de la cuenta de Voxeo 44
Figura 13. Página principal de la sección “Application Manager” 45
Figura 14. Página de creación de una nueva aplicación. 45
Figura 15. Métodos de contacto de una aplicación. 46
Figura 16. Sección para subir los ficheros de las aplicaciones. 46
Figura 17. Aspecto del depurador de Voxeo. 47
Figura 18. Programa de administración de fincas desarrollado por Laso S.L. 49
Figura 19. Arquitectura del sistema 49
Figura 20. Página de inicio del servidor 000webhost 50
Figura 21. Pantalla inicial del administrador de base de datos phpMyAdmin 52
Figura 22. Relaciones de las tablas de la base de datos 52
Figura 23. Código para la conexión a la BBDD 60
Figura 24. Organización de los módulos del sistema 60
Figura 25. Ejemplo de gramatica_comunidades.xml 61
Figura 26. Ejemplo de gramatica_claves.xml 62

Figura 27. Diagrama de secuencia del módulo Login 63
Figura 28. Escenario del módulo Login para administrador 63
Figura 29. Escenario del módulo Login para administrado 63
Figura 30. Diagrama de secuencia del módulo Comunidad (administrador) 64
Figura 31. Diagrama de secuencia del módulo Comunidad (administrado) 65
Figura 32. Escenario de uso del módulo Comunidad (administrador) 65
Figura 33. Escenario de uso del módulo Comunidad (administrado) 66
Figura 34. Ejemplo de gramatica_pisos.xml 67
Figura 35. Diagrama de secuencia del módulo Propiedades (administrador) 67
Figura 36. Diagrama de secuencia del módulo Propiedades (administrado) 68
Figura 37. Escenario de uso del módulo Propiedades (administrador) 69
Figura 38. Escenario de uso del módulo Propiedades (administrado) 69
Figura 39. Ejemplo de gramatica_cargos.xml 70
Figura 40. Diagrama de secuencia del módulo Cargos (administrador) 71
Figura 41. Diagrama de secuencia del módulo Cargos (administrado) 72
Figura 42. Escenario de uso del módulo Cargos (administrador) 72
Figura 43. Escenario de uso del módulo Cargos (administrado) 73
Figura 44. Ejemplo de gramatica_gremios.xml 74
Figura 45. Ejemplo de gramatica_proveedores.xml 75
Figura 46. Diagrama de secuencia del módulo Proveedores (administrador) 76
Figura 47. Diagrama de secuencia del módulo Proveedores (administrado) 78
Figura 48. Escenario de uso del módulo Proveedores (administrador) 78
Figura 49. Escenario de uso del módulo Proveedores (administrado) 79
Figura 50. Ejemplo de gramatica_contratos.xml 80
Figura 51. Diagrama de secuencia del módulo Contratos (administrador) 81
Figura 52. Diagrama de secuencia del módulo Contratos (administrado) 82
Figura 53. Escenario de uso del módulo Contratos (administrador) 82
Figura 54. Escenario de uso del módulo Contratos (administrado) 83
Figura 55. Código que recoge la fecha indicada por el usuario 84
Figura 56. Correo electrónico con la fecha de próxima junta 85
Figura 57. Diagrama de secuencia del módulo Juntas Vecinales (administrador) 86
Figura 58. Diagrama de secuencia del módulo Juntas Vecinales (administrado) 87
Figura 59. Escenario de uso del módulo Juntas Vecinales (administrador) 88
Figura 60. Escenario de uso del módulo Juntas Vecinales (administrado) 89
Figura 61. Código que genera gramática para el número de movimientos 90
Figura 62. Diagrama de secuencia del módulo Contabilidad (administrador) 91
Figura 63. Diagrama de secuencia del módulo Contabilidad (administrado) 93
Figura 64. Escenario de uso del módulo Contabilidad (administrador) 93
Figura 65. Escenario de uso del módulo Juntas Vecinales (administrado) 94
Figura 66. Cuestionario de evaluación 96

Índice de tablas

Tabla 1: Elementos de VoiceXML 42
Tabla 2: Atributos de la tabla Comunidad 53
Tabla 3: Atributos de la tabla Administrador 53
Tabla 4: Atributos de la tabla Admin_Comunidad 53
Tabla 5: Atributos de la tabla Propiedad 54
Tabla 6: Atributos de la tabla Propietario 54
Tabla 7: Atributos de la tabla Forma_Pago 55
Tabla 8: Atributos de la tabla Recibo 55
Tabla 9: Atributos de la tabla Recibo 55
Tabla 10: Atributos de la tabla Junta 56
Tabla 11: Atributos de la tabla Acta_Junta 56
Tabla 12: Atributos de la tabla Extracto_Contable 57
Tabla 13: Atributos de la tabla Proveedor 57
Tabla 14: Atributos de la tabla Servicio 57
Tabla 15: Atributos de la tabla Proveedor_Servicio 58
Tabla 16: Atributos de la tabla Contrato 58
Tabla 17: Resultados de la evaluación 97
Tabla 18: Resumen de costes 105

Capítulo 1

Introducción y objetivos

En este primer capítulo se hará una introducción al sistema desarrollado en el que se explicará en qué consisten tanto los sistemas de diálogo, como el ámbito de aplicación de la administración de fincas, y se marcarán los objetivos que se pretenden conseguir con la aplicación de estos sistemas de diálogo en el ámbito de la administración de fincas.

Además de esta introducción, se hará un resumen de todo el proceso que se ha llevado a cabo para el desarrollo de este Proyecto Final de Carrera. Este capítulo incluye también una explicación de las fases en las que se ha dividido el desarrollo del proyecto, una planificación temporal del tiempo necesario para este desarrollo, y una descripción de los medios requeridos. Finalmente, también se incluirá un resumen del contenido de esta memoria.

1.1 Introducción

La interacción "tradicional" con un ordenador se basa en el uso de la pantalla, el teclado y el ratón, y requiere que el usuario tenga unos conocimientos básicos acerca del funcionamiento del ordenador. Además, este tipo de interacción conlleva una barrera difícil de superar para usuarios potenciales que padecen determinadas discapacidades como pueden ser invidentes o personas con problemas de movilidad.

Para solventar estas limitaciones se están desarrollando y perfeccionando técnicas de Procesamiento del Habla e Interacción Multimodal con objeto de facilitar el uso del ordenador a todo tipo de usuarios.

La interacción multimodal pretende superar las limitaciones propias del monitor, el teclado y el ratón a la hora de interactuar con un ordenador. En este tipo de interacción el usuario puede utilizar modalidades de entrada adicionales, como por ejemplo la voz, y obtener diversas modalidades de salida con las que proporcionar mayor información al usuario.

El habla constituye una de las formas más naturales de comunicación entre las personas, de ahí el gran interés que tiene el desarrollo de sistemas informáticos capaces de procesar el habla y generarla de forma automática.

El procesamiento del habla abarca un amplio abanico de métodos y técnicas que tienen una doble finalidad. Por una parte, lograr que los ordenadores puedan comprender los mensajes pronunciados por los usuarios, y por otra, lograr que los usuarios puedan entender los mensajes generados por los ordenadores de forma oral.

Precisamente los sistemas de diálogo son, según Ramón López-Cózar "*programas informáticos que se diseñan con la finalidad de emular a un ser humano en un díalogo oral con otra persona*" [COZAR].

En los años ochenta, debido a la inexistencia de herramientas de desarrollo, para poder implementar estos sistemas de diálogo era necesario procesar a nivel de señal la voz proveniente de los usuarios, controlar el flujo de la interacción y los posibles eventos que podían producirse.

En los años noventa aparecieron diversas herramientas de desarrollo que facilitan la implementación de estos sistemas de diálogo. También aparecen en el mercado servidores web con capacidad de soportar la voz humana, propiciando la aparición de lenguajes basados en etiquetas que facilitan el desarrollo de sistemas que puedan interactuar oralmente con páginas web.

VoiceXML [VXML] es uno de estos lenguajes de etiquetas, basado en XML, que permite la interacción oral con independencia de la aplicación en la que esté implementado. De esta manera, no es necesario conocer detalles específicos de una plataforma para poder entender el funcionamiento del sistema de diálogo.

A comienzo de 2001, VoiceXML fue considerado por el World Wide Web Consortium (W3C) como el lenguaje estándar basado en etiquetas para crear los sistemas de diálogo.

1.2 Objetivos

El objetivo fundamental de este Proyecto Fin de Carrera es el de aplicar el estándar VoiceXML en un sistema de administración de fincas.

Según el artículo 20 de la Ley 49/1960, de 21 de julio [BOE60], sobre propiedad horizontal, las funciones de un administrador de fincas son las siguientes:

a) Velar por el buen régimen de la casa, sus instalaciones y servicios, y hacer a estos efectos las oportunas advertencias y apercibimientos a los titulares.

b) Preparar con la debida antelación y someter a la Junta el plan de gastos previsibles, proponiendo los medios necesarios para hacer frente a los mismos.

c) Atender a la conservación y entretenimiento de la casa, disponiendo las reparaciones y medidas que resulten urgentes, dando inmediata cuenta de ellas al presidente o, en su caso, a los propietarios.

d) Ejecutar los acuerdos adoptados en materia de obras y efectuar los pagos y realizar los cobros que sean procedentes.

e) Actuar, en su caso, como secretario de la Junta y custodiar a disposición de los titulares la documentación de la comunidad.

f) Todas las demás atribuciones que se confieran por la Junta.

Actualmente existen numerosos profesionales y empresas dedicadas a ofrecer servicios para la Administración de Fincas. Muchos de estos administradores de fincas se ayudan de programas hechos por terceros, que facilitan sus tareas diarias y ofrecen a las fincas que administran un mejor servicio. En las Figuras 1 y 2 se pueden ver algunos ejemplos de estos programas que ofrecen las empresas de administración de fincas.

Figura 1. Ejemplos de programas de administración de fincas que hay actualmente en el mercado.

Figura 2. Ejemplo de programa de administración de fincas que hay actualmente en el mercado.

Laso S.L. es la empresa de Zaragoza en la que trabajo actualmente. Esta empresa se dedica desde hace treinta años a desarrollar e innovar dentro del mundo de las nuevas tecnologías, con el fin de ayudar a las empresas, en su adaptación y mejora de las Tecnologías de la Información y Comunicaciones (TICs). Su actividad principal es el desarrollo de aplicaciones a medida, y uno de los sectores en los que trabaja es el de la administración de fincas.

El hecho de trabajar diariamente en el mantenimiento y desarrollo de un sistema de administración de fincas provocó que el Proyecto Final de Carrera se encaminara a aplicar el estándar VoiceXML en este tipo de sistemas, para poder así ofrecer al cliente una funcionalidad que actualmente no se encuentra en el mercado.

La posibilidad de tener un acceso oral a la aplicación de administración de fincas permite la comunicación de un modo rápido y sencillo en casos y lugares en los que no sería posible mediante otros medios, como puede ser en un entorno reducido o incluso desde un vehículo. Este acceso podría estar disponible en cualquier momento y además podría ser utilizado por cualquier tipo de persona, incluidas aquellas con discapacidades motoras o visuales.

1.3 Fases del desarrollo

El presente proyecto se ha dividido en tres fases, planificación, desarrollo y documentación.

Fase de planificación:

- **Estudio de los sistemas de diálogo.** Análisis de los sistemas de diálogo y el estado actual de la investigación en este campo.

- **Estudio de los sistemas de administración de fincas.** Además de los conocimientos previos de este tipo de programas gracias al trabajo realizado en la empresa Laso S.L., se han estudiado otro tipo de sistemas existentes en el mercado para conocer de esta manera con qué tipo de información se trabaja a la hora de realizar pequeñas consultas.

- **Estudio y práctica del lenguaje VoiceXML.** Ha sido necesario la consulta de manuales y la implementación de ejemplos prácticos, para conocer todo lo posible el lenguaje con el que se va a trabajar y saber todas las posibilidades que nos ofrece, así como sus limitaciones.

- **Estudio y preparación del servidor web.** Relacionar la plataforma web con el sistema de diálogo y conocer la forma de alojar todo el contenido necesario para el proyecto. Cabe destacar que, durante el desarrollo de este proyecto, una actualización del servidor seleccionado (x10Hosting) provocó numerosos problemas y finalmente se decidió cambiar de servidor (000webhost) volviendo a tener que realizar esta misma preparación.

- **Estudio de las tecnologías y lenguajes necesarios.** Además del estudio del lenguaje VoiceXML y los servidores web, que por su importancia se han mostrado como una tarea aparte, ha sido necesario el estudio de la plataforma Voxeo, el gestor de la base de datos (MySQL) y los lenguajes de programación PHP y SQL.

Fase de desarrollo:

- **Diseño de la base de datos.** Una vez que se conoce toda la información relevante de nuestro sistema se ha almacenado con una lógica que permite una fácil consulta en cualquier momento.

- **Diseño del flujo de interacción.** Se ha definido la forma en la que el usuario interactua con el sistema, así como el modo en el que se presenta y recibe la información.

- **Diseño de gramáticas.** Las gramáticas se emplean para identificar la información que el usuario transmite al sistema. Si lo que el usuario pretende comunicar no está definido en una gramática, el sistema no lo comprenderá. Se ha prestado especial atención en facilitar la interacción con el usuario de forma que sea lo más natural posible.

- **Desarrollo de la aplicación.** Se ha implementado el sistema mediante VoiceXML y lenguajes de programación como PHP y SQL. Durante cada ejecución del sistema se irán generando muchas de las gramáticas que se vayan necesitando.

- **Pruebas de evaluación y validación.** Durante el desarrollo del proyecto se han ido realizando pruebas unitarias para comprobar el correcto funcionamiento de cada uno de los módulos. Una vez todo ha funcionado correctamente se han realizado también pruebas de integración del sistema completa para alcanzar una versión completamente estable.

- **Evaluación.** Para asegurar el correcto funcionamiento del sistema se han realizado pruebas de validación con distintos usuarios con el objetivo de perfeccionar el sistema y mejorar la interactuaccion con los usuarios.

Fase de documentación:

- **Memoria del Proyecto Final de Carrera.** Redacción del presente documento, en el cual se incluye como bibliografía, toda la información que ha servido de ayuda a lo largo de todo el proyecto.

- **Preparación de la presentación.** Preparación de una explicación del contenido de este Proyecto Final de Carrera.

1.4 Planificación temporal

Una vez establecidas las fases y tareas en las que se ha dividido el proyecto, se ha realizado una planificación temporal de estas fases.

Debido al trabajo a tiempo completo que se ha ejercido durante el desarrollo de este proyecto y el hecho de no tener una fecha límite para la entrega cercana en el tiempo, ha provocado que el tiempo dedicado a cada tarea sea muy intermitente y que las tareas se prolonguen durante mucho tiempo. Por esta intermitencia se ha desestimado la utilización del *diagrama de Gantt* para esta planificación temporal, y el tiempo, en horas, planteado para cada tarea se muestra listado a continuación.

Fase de planificación:

- **Estudio de los sistemas de diálogo.**
  Duración: 20 horas.

- **Estudio de los sistemas de administración de fincas.**
  Duración: 10 horas.

- **Estudio y práctica del lenguaje VoiceXML.**
  Duración: 80 horas.

- **Estudio y preparación del servidor web.**
  Duración: 10 horas.

- **Estudio de las tecnologías y lenguajes necesarios.**
  Duración: 20 horas.

Fase de desarrollo:

- **Diseño de la base de datos.**
  Duración: 40 horas.

- **Diseño del flujo de interacción.**
  Duracion: 40 horas.

- **Diseño de gramáticas.**
  Duracion: 60 horas.

- **Desarrollo de la aplicación.**
  Duracion: 240 horas.

- **Pruebas.**
  Duracion: 120 horas.

- **Evaluación.**
  Duracion: 20 horas.

Fase de documentación:

- **Memoria del Proyecto Final de Carrera.**
  Duracion: 100 horas.

- **Preparación de la presentación.**
  Duracion: 20 horas.

DURACIÓN TOTAL: 800 horas.

1.5 Medios empleados

Los medios utilizados para llevar a cabo el Proyecto Final de Carrera son los siguientes:

- **Recursos software:**
  - Mozilla Firefox.
  - Plataforma Voxeo.
  - Skype.
  - Microsoft Office 2016.

  - EditPlus Text Editor.

- **Recursos hardware**
  - Ordenador portátil.
  - Periféricos habituales (teclado, ratón).
  - Impresora HP Deskjet 3637.
  - Auriculares con micrófono.
  - Servidor 000webhost.
  - Router.

En el apartado Presupuesto de esta memoria se incluye la información referente al coste de estos medios.

En cuanto a la documentación empleada, se han examinado numerosos libros, artículos y recursos web sobre los sistemas de diálogo y lenguajes de computación oral, así como manuales de los distintos lenguajes de programación utilizados. Toda esta documentación se encuentra detallada en el apartado de Bibliografía.

1.6 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo.

Capítulo 1: Introducción. Este primer capítulo muestra las motivaciones y objetivos del Proyecto Final de Carrera. Además, incluye las fases de desarrollo, la planificación temporal, los medios empleados y la descripción de la estructura de esta memoria.

Capítulo 2: Estado del Arte. En este capítulo se hace un estudio detallado de los sistemas de diálogo, se analiza en detalle el lenguaje VoiceXML y se presenta la plataforma utilizada para la implementación de la aplicación, Voxeo.

Capítulo 3: Descripción general del sistema desarrollado. En este capítulo se introducirá el sistema creado explicando su funcionalidad y arquitectura. Además, se analizará el servidor web empleado y la base de datos utilizada.

Capítulo 4: Explicación detallada del sistema desarrollado. En esta sección se describe detalladamente cada uno de los módulos de los que consta el sistema desarrollado para la administración de fincas: Login, Comunidad, Propiedades, Cargos, Proveedores, Contratos, Juntas Vecinales y Contabilidad. Para cada uno de ellos se explica sus funcionalidades, arquitectura y escenarios de uso.

Capítulo 5: Evaluación del sistema. En este capítulo, a través de las valoraciones subjetivas de los usuarios recogidas mediante un cuestionario, se lleva a cabo la evaluación del sistema desarrollado.

Capítulo 6: Conclusiones y trabajo futuro. Se exponen las principales ideas, cuestiones y conclusiones derivadas de la realización del proyecto, así como las posibles líneas de investigación que a partir de este proyecto se podrían generar.

Capítulo 7: Presupuesto. Este apartado contiene un análisis de los costes del diseño y desarrollo del proyecto, detallando el coste de personal y del material necesario para llevar a cabo su realización.

Glosario. Se ofrece una recopilación de los principales términos y conceptos técnicos empleados en la memoria, con el objetivo de facilitar su comprensión al lector.

Bibliografía. En este apartado están reflejadas las citas bibliográficas que se han consultado para la realización del proyecto y la memoria.

Capítulo 2

Estado del Arte

En este capítulo se mostrará una visión global indicando cómo se enmarca la aplicación desarrollada y qué aporta frente al resto del estado del arte.

Partiendo de esta idea, es necesario establecer las bases que nos permitan comprender cuál es la situación en la que se encuentran actualmente los sistemas de diálogo, analizando sus módulos y procedimientos mediante los que interactúan entre sí y qué aplicaciones principales hacen uso de ellos. A continuación, se resumirán las principales características del lenguaje VoiceXML. Para terminar, se describirá Voxeo, intérprete VoiceXML utilizado para el proyecto y que permite asignar un número de teléfono a la aplicación y disponer de un servidor para hospedar el código desarrollado en este lenguaje.

2.1 Sistemas de Diálogo

2.1.1 Introducción

Los sistemas de diálogo basados en procesamiento del habla son sistemas informáticos que reciben como entrada frases del lenguaje natural expresadas de forma oral y generan como salida frases del lenguaje natural expresadas asimismo de forma oral [LOP05]. Un sistema de diálogo puede, por lo tanto, entenderse como un sistema

automático capaz de emular a un ser humano en un diálogo con otra persona, con el objetivo de que el sistema cumpla con una cierta tarea (normalmente suministrar una cierta información o llevar a cabo una determinada tarea) [GRI07].

Un sistema de diálogo ideal reconocería el habla espontánea, comprendería enunciados sin restricciones de contenido, proporcionaría respuestas con sentido, gramaticalmente bien formadas y pragmáticamente adecuadas, respondería con voz completamente natural y sería multimodal [LLI06].

Las limitaciones o restricciones que suelen presentarse en estos sistemas son las siguientes [LLI06]:

- Se encuentran sujetos a las limitaciones en lo que respecta al reconocimiento automático del mensaje oral.
- Tanto la comprensión como la posible respuesta están restringidas a dominios específicos.
- Existe un condicionamiento claro debido a la naturalidad del habla sintetizada.
- Tienen una evidente dependencia de la necesidad de implementar estrategias de verificación.
- No pueden abstraerse de los problemas propios del diálogo espontáneo: elipsis, anáforas, deícticos.

Definidas estas limitaciones, hay que destacar también cuáles son las capacidades que un sistema de diálogo debe contemplar para ser capaz de comportarse de una manera análoga a la humana. Estas capacidades son [LLI06]:

- Habilidad para el reconocimiento de las elocuciones del usuario.
- Inclusión de una herramienta para la gestión del diálogo.
- Capacidad para el análisis lingüístico (morfológico, sintáctico, semántico, pragmático) de los enunciados.
- Creación de una representación interna de la historia del diálogo.
- Mecanismo para el tratamiento de la representación interna en función de la tarea.
- Implementación de un sistema para la generación de secuencias de respuesta.
- Herramienta de conversión del texto en lenguaje oral.

2.1.2 Arquitectura de un sistema de diálogo

Los sistemas de diálogo tienen como finalidad que un usuario, a menudo a través del teléfono o de un micrófono y sin un intermediario humano, pueda acceder automáticamente a una determinada información, realizar una transacción o conseguir la ejecución de determinadas órdenes. Dado que el medio utilizado es la lengua oral, se requiere la integración de diferentes componentes, como se puede observar en la Figura 3 [LLIS03].

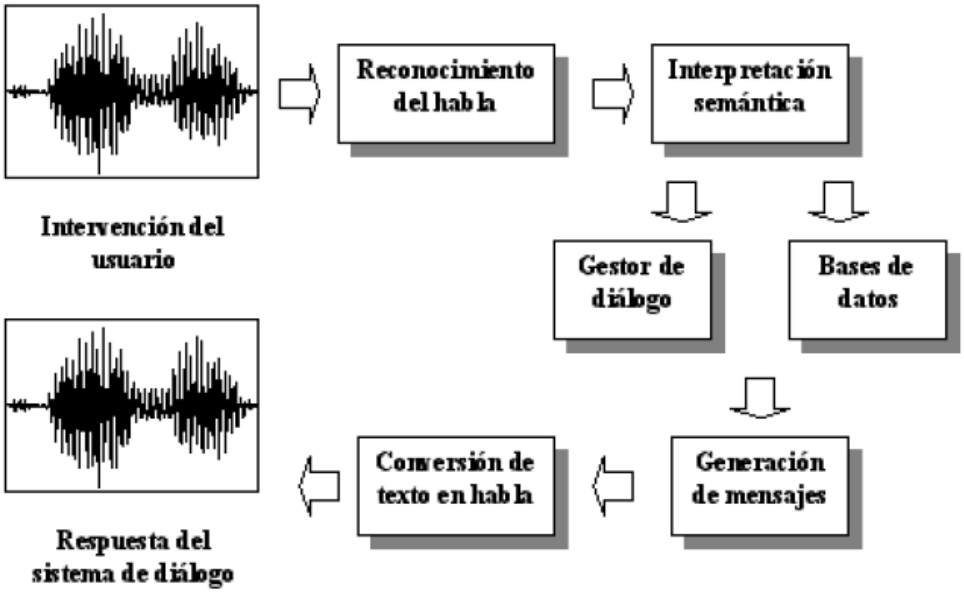

Figura 3. Arquitectura de un sistema de diálogo

Cada uno de los módulos que forman parte de un sistema de diálogo tiene una función específica:

Reconocimiento del habla

Procesa el mensaje recibido por el usuario y lo transforma en una secuencia de palabras que pueden ser interpretadas semánticamente con más facilidad.

Los sistemas de reconocimiento automático de habla han mejorado mucho en los últimos años gracias al desarrollo de mejores algoritmos y modelos acústicos, y a la aparición de procesadores con mayor capacidad de cálculo. Esta evolución ha conseguido que los sistemas de reconocimiento de voz hayan pasado de ser sistemas discretos que reconocían palabra por palabra y número por número, a ser ahora sistemas continuos y naturales capaces de procesar una respuesta a una velocidad de conversación normal del usuario. Un esquema de este tipo de sistema se muestra en la Figura 4.

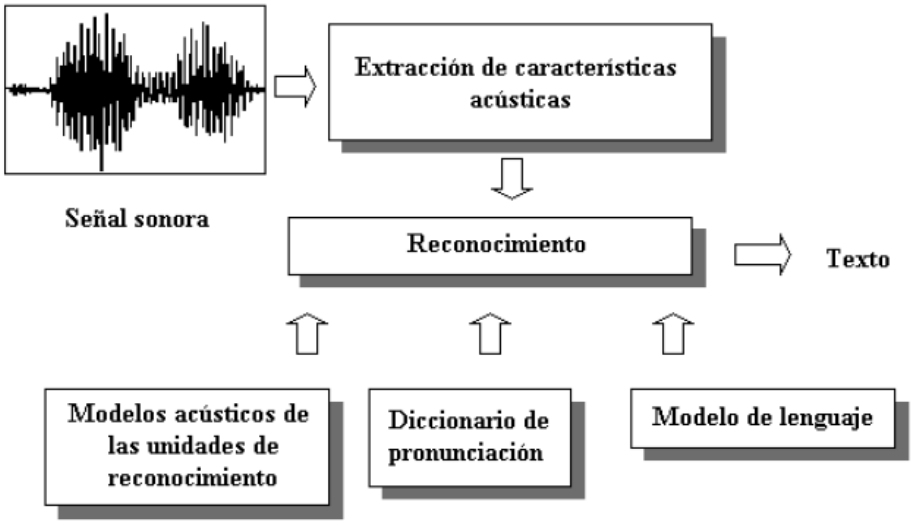

Figura 4. Esquema de un sistema de reconocimiento del habla

Interpretación semántica

El sistema analiza la intervención del usuario y la interpreta. El descodificador o módulo de interpretación semántica debe ser capaz de resolver el significado de un determinado tipo de expresiones relacionadas con la aplicación que va a tener ese sistema.

Gestión de diálogo

Decide qué paso debe dar el sistema tras cada intervención del usuario. Puede decirse que este es el módulo fundamental del sistema, pues su finalidad es lograr que la interacción con el usuario sea lo más cómoda y sencilla posible. Para ello, se basa en la interpretación semántica de la petición del usuario, la historia del proceso de diálogo, la información disponible en ese punto, el estado actual del sistema, la información obtenida de la base de datos, la estrategia definida, etc.

Bases de datos

Una vez que el sistema ha identificado lo que desea el usuario, accede, si es preciso, a los repositorios de datos y genera un mensaje de respuesta adecuado.

Generador de mensajes

Tiene como función la generación de una frase, gramaticalmente correcta y en un lenguaje lo más cercano posible al lenguaje natural, que transmita el mensaje generado por el gestor de diálogo.

Sintetizador de texto en habla

Es el encargado de proporcionar un mensaje oral al usuario, de manera que éste acabe obteniendo una respuesta adecuada a sus necesidades. En la Figura 5 se puede observar un esquema del funcionamiento de este tipo de sintetizador.

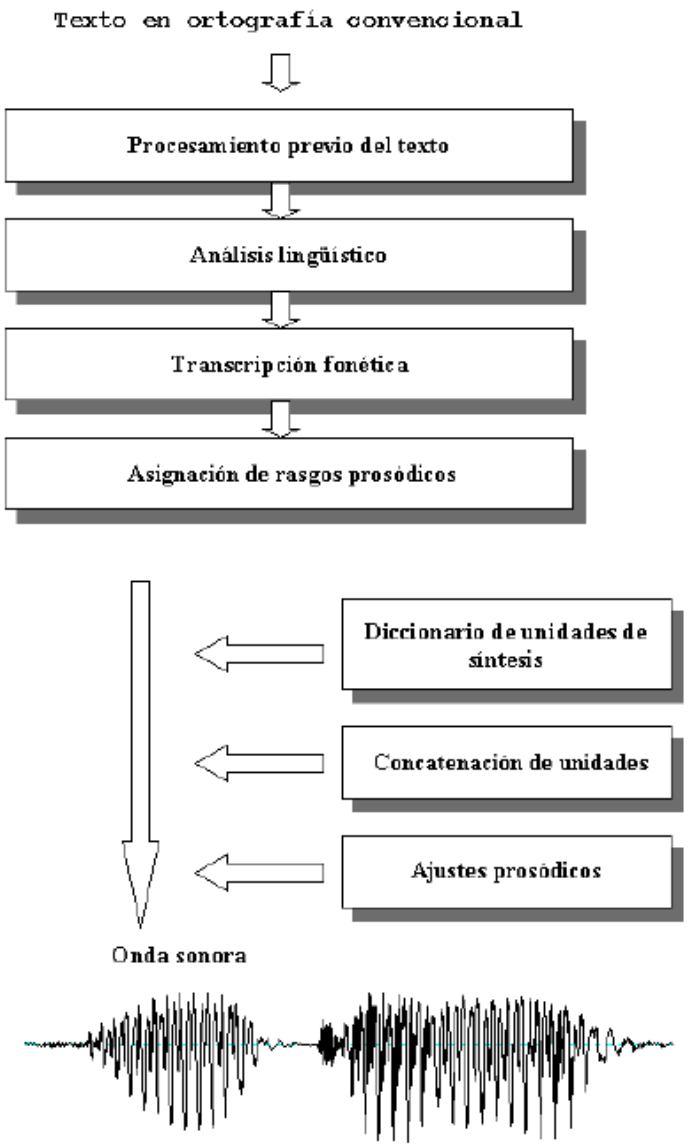

Figura 5. Esquema del procesamiento lingüístico en un sintetizador de texto en habla [LLIS03]

2.1.3 Aplicaciones de los sistemas de diálogo

En la actualidad existen numerosas empresas y grupos de investigación que ofrecen sistemas de diálogo para proporcionar información de formas automática al usuario.

Una de las empresas punteras en este ámbito en España es Natural Vox, una empresa de Vitoria que lleva más de 20 años trabajando en el desarrollo de sistemas de telefonía automática. [NATURALVOX]

En la actualidad los servicios que ofrece Natural Vox se engloban en 4 productos:

- **Banca por teléfono (BPT).** Da servicio a todas las llamadas de los clientes 24 horas al día, 7 días por semana. Destaca por la naturalidad, sin limitar las posibilidades de

lo que pueda pedir el usuario y automatizando las operaciones más complejas: traspasos, transferencias, fondos de inversión (suscripción y reembolso), valores, extractos por fax, etc.

- **Sistemas de información**. Los sistemas de información telefónica de pueden contener cientos de informaciones diferentes que incluyan las solicitudes más frecuentes de los usuarios que llamen por teléfono.

- **Cita previa.** No más colas ni esperas para los clientes a la hora de coger una cita. Permite trabajar con personal para facilitar aún más el acceso al sistema, o bien dejarlo en modo automático con lo que el sistema se basa en reconocimiento discreto y continuo para concertar una cita durante las 24 horas de los 365 días del año.

- **Venta de billetes y entradas.** En la actualidad el número de operaciones realizadas a través del teléfono es cada vez mayor, llegando incluso a realizarse compras utilizando este canal. Este servicio ha creado también un sistema de ticketing en el que el usuario además de comprar su entrada o billete, puede reservar o pedir información sobre las tarifas, horarios, fechas, disponiblidad, puntos de venta, etc.

En la Figura 6 se muestran algunos de los clientes de NaturalVox, agrupados por el tipo de servicio que utilizan.

Figura 6. Algunos clientes y aplicaciones de la empresa Natural Vox.

Además de los servicios ofrecidos por Natural Vox, existen otras muchas aplicaciones de los sistemas de diálogo. Algunas de las más destacadas son las siguientes:

- **Alexa**. Asistente virtual de Amazon que pretende hacer más fácil la vida cotidiana en el hogar [ALEXA].

- **Siri.** Asistente de Apple capaz de responder a preguntas directas hacer recomendaciones y realizar acciones mediante la delegación de solicitudes hacia un conjunto de servicios web que ha ido aumentando con el tiempo.

- **Irene.** Asistente de Renfe disponible en varios idiomas que ayuda a preparar los viajes [IRENE].

- **BusLine.** Aplicación de VoiceXML construida con la plataforma TellMe que permite conocer los autobuses alrededor de CMU, Oakland y Squirrel Hill [BUSLINE].

- **Jupiter.** Es un sistema de conversación que proporciona información metereológica actualizada por teléfono [JUPITER].

- **Let's Go!** Sistema de diálogo oral que proporciona acceso a la ruta de autobuses e información de la programación en el área de Pittsburgh [LETSGO].

- **Mercury.** Interfaz de conversación telefónica que proporciona información sobre horarios de vuelos y precios. Mercury permite a los usuarios reservar y comparar itinerarios complejos de viajes a más de 200 ciudades con Estados Unidos y alrededor del mundo [MERCURY].

- **Pegasus.** Interfaz de conversación que proporciona información a través de una línea telefónica sobre el estado del vuelo en Estados Unidos [PEGASUS].

- **Listen.** Tutor de lectura automatizado que muestra historias en una pantalla de ordenador y escucha a los niños leer en voz alta. El Tutor de Lectura interviene cuando el lector comete errores, se queda atascado, hace clic para obtener ayuda o cree que encuentre dificultades [LISTEN].

- **Universal Speech Interface.** Interfaz de voz universal que permite a las personas comunicarse eficazmente, eficientemente y sin esfuerzo con máquinas simples y servicios de información. Algunos de los servicios en los que está implementado son: *MovieLine:* un servicio interactivo telefónico que permite a los usuarios conocer películas y salas de cine en el área de Pittsburgh. *ApartmentLine*: un servicio similar con información sobre alquiler de vivienda. *FlightLine*: salidas de vuelo, llegadas e información de la puerta. *Gadget*: control inalámbrico de gadgets y dispositivos [USI].

- **Voyager.** Sistema de conversación que puede participar en diálogos verbales con los usuarios sobre información turística y de viaje para el área del Gran Boston [VOYAGER].

- **Waxholm.** Sistema que ofrece información sobre el tráfico de barcos en el archipiélago de Estocolmo [WAXHOLM].

2.2 Voice Extensible Markup Language (VoiceXML)

2.2.1 Introducción al estándar VoiceXML

El lenguaje VoiceXML tuvo sus orígenes en 1995. Fue diseñado como un lenguaje de diálogo basado en XML (eXtensible Markup Language) [XML], que pretendía simplificar el proceso de desarrollo de aplicaciones de reconocimiento de voz dentro de un proyecto de AT&T llamado PML (Phone Markup Language).

En 1998, el W3C (Consorcio World Wide Web) organizó una conferencia sobre navegadores por voz. En este momento, AT&T y Lucent tenían diferentes variantes de su PML original, mientras que Motorola e IBM habían desarrollado sus propios lenguajes VoxML y SpeechML respectivamente. Muchos otros asistentes a la conferencia también estaban diseñando lenguajes parecidos, por ejemplo, TalkML de HP y VoiceHTML de PipeBeach.

Debido a esta situación, AT&T, IBM, Lucent y Motorola decidieron formar el Foro de VoiceXML para aunar sus esfuerzos. La misión del Foro VoiceXML era definir un lenguaje estándar que los desarrolladores pudieran utilizar para crear aplicaciones de voz. Eligieron XML como base para ello.

En 2000, el Foro de VoiceXML lanzó VoiceXML 1.0 al público. Poco después, VoiceXML 1.0 se presentó al W3C como base para la creación de un nuevo estándar internacional.

Finalmente, VoiceXML 2.0 es el resultado de ese trabajo basado en la colaboración conjunta de las empresas miembros de W3C, otros grupos de trabajo del W3C y el público. La versión 2.1 aparece como recomendación del W3C en junio de 2007. El working draft de VoiceXML 3.0 fue aprobado en diciembre de 2010.

La utilización del estándar promueve la integración de los sistemas de diálogo, así como el aseguramiento de su portabilidad y la construcción de herramientas que faciliten al usuario la definición de servicios de diálogo.

Las estructuras de estos servicios se definen a partir de entradas, que pueden consistir en gramáticas habladas o tonos DTMF (Dual-tone multi-frequency) y salidas en formato de conversores texto-a-voz o grabaciones de audio.

La secuencia de interacción entre las entradas y las salidas se definen en VoiceXML.

VoiceXML es un lenguaje de etiquetas basado en XML que permite describir servicios de voz con independencia de la aplicación en la que corran. De esta manera no

es necesario conocer detalles específicos de una plataforma para entender el funcionamiento del sistema de diálogo.

El lenguaje VoiceXML describe la interacción hombre-maquina a partir de los siguientes elementos:

- Salida de texto-a-voz
- Salida de audio grabado
- Reconocimiento de entrada hablada
- Reconocimiento de tonos DTMF
- Grabación de entrada hablada
- Control de flujo de diálogo
- Funciones de telefonía (transferencia de llamada, desconexión, etc.).

Mientras HTML basa su funcionalidad en un navegador gráfico con pantalla, teclado y ratón, VoiceXML funciona mediante un navegador de voz cuya salida es audio, y cuya entrada es audio y teclado. La entrada de audio está controlada por un reconocedor de voz integrado con el navegador de VoiceXML. La salida de audio consiste en audio pregrabado y/o en voz sintetizada por un sistema de Text-To-Speech.

Un navegador de voz normalmente funciona en base a una pasarela de voz que es un nodo conectado tanto a Internet como a la RTC (Red Telefónica Conmutada). La pasarela de voz puede soportar cientos o miles de llamadas simultáneas y permite que accedan a ella cualquiera de los casi 9.000 millones de teléfonos que se estima que hay en el mundo, entre los terminales móviles y los teléfonos fijos. [VERBIO]

La Figura 7 resume la arquitectura descrita.

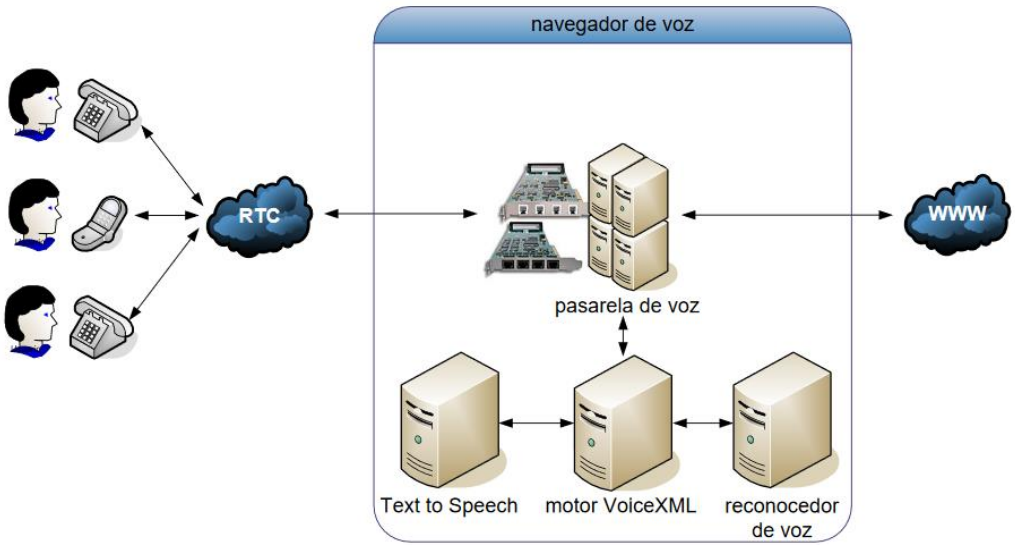

Figura 7. Arquitectura de un sistema VoiceXML

La descripción de un servicio básico de diálogo tendría el aspecto mostrado en la Figura 8.

```
<?xml version="1.0" encoding="UTF-8"?>
    <vxml xmlns="http://www.w3.org/2001/vxml"
    xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xsi:schemaLocation="http://www.w3.org/2001/vxml http://www.w3.org/TR/voicexml20/vxml.xsd" version="2.0">

    <form>

        <field name="libro">
            <prompt>¿Quiere escuchar Caperucita, Blancanieves o Cenicienta? </prompt>

            <grammar src="libro.grxml" type="application/srgs+xml"/>
        </field>
        <block>
            <submit next="http://www.library.com/libro2.asp"/>
        </block>
    </form>

</vxml>
```

Figura 8. Ejemplo de un programa de VoiceXML

Este ejemplo sencillo implementaría un servicio de libros leídos donde el usuario ha de escoger entre escuchar el cuento de Caperucita, de Blancanieves o de Cenicienta. El campo *field* es un equivalente al campo tipo input del lenguaje HTML sólo que en este caso la entrada es hablada en lugar de tecleada. La gramática de entrada se definiría siguiendo el estándar SRGS (*Speech Recognition Grammar Specification*) [SRGS].

Si el sistema reconoce las palabras Caperucita, Blancanieves o Cenicienta, reproduciría un archivo de sonido con la versión en audio del cuento seleccionado.

La aplicación consistiría en interprete de documentos VoiceXML que interactúa con una plataforma de voz, que en este caso puede ser un sistema de telefonía recogiendo las instrucciones habladas de un usuario o alternativamente la marcación por pulsos DTMF, como haría un sistema clásico IVR (Interactive Voice Response).

Podríamos resumir en 4, los motivos por los cuales es tan importante la penetración y el enfoque de la tecnología VoiceXML [VERBIO]:

- En primer lugar, porque el teléfono es una herramienta con mucho poder. Actualmente hay más de 9.000 millones de teléfonos en uso. Una cifra muy superior a la de usuarios que disponen de ordenador con conexión a Internet. Además, los teléfonos son fáciles de usar y no necesitan ni tiempo de arranque ni sistemas operativos.

También los teléfonos móviles están consiguiendo una gran penetración en la población. A diferencia de los ordenadores portátiles y algunas PDAs, los teléfonos móviles son fáciles de llevar, no demasiado caros y tienen baterías con un tiempo de vida largo. Además, pueden usarse mientras se conduce o se tienen las manos ocupadas.

- En segundo lugar, porque la voz es importante en el teléfono. La voz siempre ha sido la forma natural de comunicarse a través del teléfono. Aunque algunos móviles disponen de navegadores WAP o XHTML, sus pequeñas pantallas y teclados, hacen

complicada la navegación, especialmente en algunas circunstancias como por ejemplo durante la conducción.

Sin embargo, existen ventajas al combinar navegadores visuales con navegadores de voz. Por ejemplo, la información compleja es difícil de recordar cuando se habla con el usuario, pero sencilla de recordar si esta persistente de forma visual. Y algunos fallos de reconocimiento son fáciles de corregir con una entrada de teclado. En consecuencia, es previsible que con el tiempo aparezcan aplicaciones multimodales, desarrolladas de la unión de aplicaciones estrictamente visuales y aplicaciones estrictamente de voz.

- En tercer lugar, porque Internet ayuda al desarrollo de aplicaciones de voz en VoiceXML.

El desarrollo de aplicaciones de voz es más sencillo debido a que VoiceXML es un lenguaje de marcas de alto nivel y porque las aplicaciones de voz ahora se pueden construir en su plenitud con herramientas potentes, sin un coste excesivo.

Las aplicaciones de voz son ahora más que nunca, sencillas de distribuir. Ya no es necesario que residan en servidores de voz dedicados a esta tarea: pueden alojarse en cualquier sitio y accedidas por cualquier servidor de VoiceXML.

Las aplicaciones pueden estructurarse de forma ordenada y limpia, entregando distintas páginas VoiceXML al navegador de voz en función de por ejemplo el dispositivo que las está navegando.

- Finalmente, la voz y VoiceXML son importantes para otros dispositivos Web además del teléfono. Por ejemplo, un “control remoto universal”, podría llevar un navegador VoiceXML incorporado o comunicarse con alguno mediante tecnología Wireless. Podrías entrar en el comedor, sacar el control remoto, pulsar el botón de hablar y decir algo como: “equipo de música: apagar, televisión: ¿qué películas de acción hacen hoy?”

2.2.2 Ventajas e inconvenientes de VoiceXML

VoiceXML es una tecnología independiente de la plataforma. Las herramientas construidas y adaptadas para soportar el estándar comparten un formato de definición de sistemas de diálogo que permite la portabilidad y transferencia de datos entre aplicaciones heterogéneas. Además, este tipo de herramientas tienden a facilitar enormemente el trabajo del desarrollador, que no necesita conocer los detalles de implementación del sistema que está construyendo.

No obstante, el seguimiento del estándar produce también inconvenientes. VoiceXML implementa funciones de telefonía, pero el conjunto de funciones es demasiado reducido y su funcionalidad muy limitada. Por ello los sistemas desarrollados con VoiceXML no consiguen alcanzar la complejidad necesaria para algunos flujos de telefonía y operaciones de voz.

En lo referente a las carencias en servicios de telefonía, W3C propone el estándar CCXML (*Call Control eXtensible Markup Language* [CCXML]), que se puede utilizar de manera conjunta con VoiceXML y que permite definir una lógica más compleja y completa del flujo telefónico, incorporando elementos necesarios como manejadores de eventos u objetos de conferencia.

Asimismo, encontramos otros estándares del W3C a tener en cuenta en la construcción de una herramienta para implementaciones de sistemas de diálogo como SRGS o SSML (*Speech Synthesis Markup Language* [SSML]).

Como vemos el W3C no sólo compensa los inconvenientes que pueda presentar VoiceXML mediante la propuesta del uso conjunto de estándares, sino que está en un proceso de permanente diálogo con la comunidad de desarrolladores para introducir ampliaciones y mejoras al estándar.

Entre las características más llamativas de VoiceXML 3.0, podemos mencionar la identificación de voz, que utiliza medidas biométricas para asegurar la identidad en transacciones telefónicas y comunicaciones.

2.2.3 Aplicaciones adecuadas de VoiceXML

Algunas de las aplicaciones más adecuadas para el uso de VoiceXML son las siguientes.

Sistemas de Información:

Los sistemas de información son un ejemplo tipo de aplicación VoiceXML. En un sistema de información, la salida de audio tiende a ser audio pregrabado y la entrada de información pueden ser comandos de voz bastante “cerrados” (comandos de navegación, fechas, respuestas si/no, etc.) o respuestas más naturales (direcciones arbitrarias de calles, etc.).

Un buen ejemplo de un sistema de información es por ejemplo una aplicación donde primero el usuario se personaliza un boletín informativo en un sitio Web y posteriormente llama periódicamente para navegar a través de las noticias del boletín. El boletín puede contener noticias de última hora, deportes, información del tráfico, del tiempo, de la bolsa, etc. El servicio podría ganar dinero a base de suscripciones, anuncios o tiempo de conexión.

Comercio Electrónico:

El comercio electrónico es otra área con gran cabida para VoiceXML. Aplicaciones para ofrecer servicios de tracking de pedidos, estados de cuenta o soporte, son algunos ejemplos.

También tienen cabida las aplicaciones financieras como, por ejemplo, consulta de acciones (bolsa), administración de tareas o aplicaciones catálogo. Las aplicaciones de catálogos deben estar hechas minuciosamente ya que la voz, en estos casos ofrece menos información que las imágenes. Estas aplicaciones pueden funcionar por ejemplo si el cliente esta mirando simultáneamente un catálogo impreso (p.ej. de ropa) o conoce exactamente a priori el producto (ej.: un libro, un CD o un DVD).

Servicios telefónicos

Los servicios telefónicos como por ejemplo los servicios de marcación por voz, de búsqueda de números telefónicos, de administración de correo electrónico o teleconferencia, pueden ser fácilmente utilizados mediante voz a través de VoiceXML. Las aplicaciones personales de voz, añadidas a las líneas particulares de los usuarios, pueden ser importantes fuentes de ingreso.

Ya que las medidas de seguridad estándar de la Web aplican también a la tecnología de voz en la Web, las aplicaciones de intranet pueden ser también programadas en VoiceXML para el control de inventario, entrega de pedidos, servicios de recursos humanos o atención al cliente, o para crear portales de voz corporativos.

Mensajería unificada:

Las aplicaciones de mensajería unificada pueden verse impulsadas gracias a la voz. Los mensajes de voz pueden leerse a través del teléfono, los mails de salida pueden grabarse (y en el futuro quizá transcribirse) y enviarse como un fichero de voz grabado, y la información relativa a direcciones, sincronizarse con agendas personales y sistemas de email.

Hay muchas otras áreas donde los servicios de voz pueden ser de gran utilidad, como las aplicaciones de directorio asistidas, verificación del estado de apuestas, autorización de pagos, programación de un servicio despertador de un hotel, etc.

2.2.4 Conceptos básicos de VoiceXML

Un documento en VoiceXML puede entenderse como una máquina de estado conversacional. Es decir, el usuario está siempre en un estado de diálogo constante con la computadora. Cada diálogo determina la transición al diálogo siguiente. Estas transiciones se especifican mediante URIs (*Uniform Resource Identifier*), que definen el siguiente documento y diálogo a utilizar. La ejecución termina cuando un diálogo no especifica un sucesor, o si se introduce algún elemento que provoque una salida explícita de la conversación.

Algunos de los conceptos básicos de VoiceXML son los siguientes:

Diálogos y Subdiálogos

Existen dos tipos de diálogos: formularios y menús.

- Los formularios son los componentes más importantes de un documento VoiceXML. Contienen los campos de entrada y de control, la declaración de variables, el tratamiento de eventos y las acciones a ejecutar cuando se completen los campos de entrada.

Los formularios son ejecutados con el algoritmo FIA (*Form Interpretation Algorithm*). Mediante este algoritmo, se almacena el contenido de los campos cuando se cumplan las condiciones correspondientes, reproduce los *prompts* necesarios y comprueba si la condición <*filled*>, asociada al formulario, se cumple.

Los formularios pueden ser de tipo directo o de iniciativa mixta. En los primeros, se recorren los campos de forma secuencial, mientras que los segundos usan gramáticas externas, pudiendo rellenarse los campos en cualquier orden.

Cada campo del formulario tiene una variable asociada, que podrá ser accedida durante el resto del diálogo, y cuyo valor será la interpretación del formulario.

- Los menús son formularios simplificados en el que el usuario debe seleccionar entre un conjunto de opciones y se determina las transiciones a nuevos documentos en función de la opción seleccionada.

Un subdiálogo es como una llamada a función, que proporciona un mecanismo para invocar una nueva interacción y volver al formulario original. Las instancias de variables, las gramáticas y la información de estado se guardan y están disponibles al volver al documento inicial.

Cuando los subdiálogos son invocados añaden un nuevo contexto de ejecución. El subdiálogo puede ser un nuevo diálogo dentro del documento existente o un nuevo diálogo dentro de un nuevo documento. Los subdiálogos pueden estar compuestos de varios documentos.

Gramáticas

Son el mecanismo mediante el cual VoiceXML permite introducir un modelo de lenguaje. Una gramática determina una secuencia de palabras aceptable durante el reconocimiento del habla. El usuario debe mencionar alguna de las expresiones determinadas por la gramática, y en función de esta mención se ejecutará una acción determinada o se proporcionará una información específica.

Las gramáticas pueden ser o externas. En caso de que existan varias gramáticas en el formulario, se puede establecer una jerarquía entre ellas especificando el peso de cada una.

Las gramáticas internas se activan únicamente cuando el algoritmo procesa el campo correspondiente. El ámbito de las gramáticas externas (accedidas con enlace) es el del elemento que engloba dicho enlace. Para las gramáticas asociadas a un formulario, el ámbito puede ser el del formulario, el del documento o el de la aplicación completa (si estuvieran englobadas en el documento ROOT). Los menús también pueden tener asociadas gramáticas, en cuyo caso, el ámbito será el del diálogo correspondiente.

Cuando al procesar un campo, el intérprete está a la espera de una entrada por parte del usuario, las gramáticas activas son:

1. Gramáticas asociadas a dicho campo.
2. Gramáticas del formulario.
3. Gramáticas contenidas en enlaces dentro del documento.
4. Gramáticas contenidas en el documento ROOT.
5. Gramáticas definidas en la captura de eventos de plataforma (ayuda, salida, cancelar).

El orden es el mostrado, y en caso de haber más de una gramática activa, la preferencia vendría determinada por el orden de los documentos. Como mínimo, cuando se espera la respuesta del usuario, debe haber activa al menos una gramática.

Sesión

Es el proceso que comienza cuando el usuario comienza a interactuar con el intérprete de contexto VoiceXML, continúa mientras los documentos son cargados y procesados, y finaliza cuando el usuario así lo requiere, aunque también es posible que sea un documento o el intérprete de contexto el que ponga fin a la sesión.

Aplicación

Una aplicación es un conjunto de documentos que comparten el mismo documento raíz de la aplicación. Siempre que el usuario interactúa con un documento en una aplicación, el documento raíz de la aplicación se carga también y permanece cargado mientras el usuario se encuentra en transición entre documentos dentro de la misma aplicación. Finalmente se descarga cuando el usuario pasa a un documento que no está en la aplicación.

Cuando una aplicación tiene un solo documento, la ejecución del documento comienza por defecto en el primer diálogo. Según se ejecuta cada diálogo, se determina el siguiente diálogo. Cuando el diálogo no especifica un diálogo sucesor, se detiene la ejecución del documento.

Si una aplicación tiene varios documentos, cada documento se ejecuta como una aplicación aislada. En los casos en los que se quiera tener varios documentos para trabajar conjuntamente como una única aplicación, se debe seleccionar un documento como documento raíz de la aplicación y el resto como documentos hoja. Cada documento hoja nombra al documento raíz en su elemento *<vxml>*.

Cada vez que el intérprete tenga que cargar y ejecutar un documento hoja, primero cargará el documento raíz de la aplicación si no está ya cargado. El documento raíz de la aplicación permanecerá cargado hasta que el intérprete tenga que cargar un documento que pertenezca a otra aplicación. Por lo tanto, siempre se mantendrá una de las dos condiciones siguientes durante la interpretación:

- El documento raíz de la aplicación se carga y el usuario está ejecutando en él. No hay ningún documento hoja.

- El documento raíz de la aplicación y un sólo documento hoja son cargados. El usuario está ejecutando en el documento hoja.

Si hay una cadena de subdiálogos definida en documentos separados, puede haber más de un documento hoja cargado, aunque la ejecución se realizará sólo en uno de estos documentos.

Cuando un documento hoja hace que se cargue un documento raíz, ninguno de los diálogos en el documento raíz se ejecutan. La ejecución comienza en el documento hoja.

Una de las ventajas de las aplicaciones multidocumento es que las variables y las propiedades del documento raíz están disponibles para su uso por los documentos hoja, de esta forma la información se puede compartir y conservar.

Eventos

VoiceXML proporciona un mecanismo de rellenado de formularios para el manejo de entradas de usuario "normales" y define un mecanismo para controlar los eventos no cubiertos por este procedimiento.

Los eventos son lanzados por la plataforma debido a una gran variedad de circunstancias, como por ejemplo, cuando el usuario no responde, no responde correctamente, solicita ayuda, etc. El intérprete también produce eventos si encuentra un error semántico en un documento de VoiceXML. El control de eventos comunes puede especificarse en cualquier nivel, y se aplica a todos los niveles inferiores.

Enlaces

Los enlaces permiten especificar gramáticas, así como transferir el control a otros documentos VoiceXML.

2.2.5 Elementos de VoiceXML

En la siguiente tabla se especifican los diferentes elementos de VoiceXML

ELEMENTO	FUNCIONALIDAD
<assign>	Asigna un valor a una variable
<audio>	Reproduce un fichero de audio
<block>	Contenedor para el contenido ejecutable
<break>	Designa una pausa

<catch>	Captura un evento
<choice>	Define las opciones disponibles de un menú
<clear>	Borra el contenido de una variable
<data>	Obtiene contenido de un código XML
<disconnect>	Desconecta una sesión
<else>	Elemento final de una sentencia condicional
<elseif>	Especifica el contenido cuando los demás elementos condicionales son falsos
<emphasis>	Especifica el nivel de un texto
<enumerate>	Enumera las opciones de un menú
<error>	Captura un evento de error
<example>	Define una frase de ejemplo
<exit>	Sale de una sesión
<field>	Declara la entrada de un campo en un formulario
<filled>	Especifica la acción a realizar tras una gramática
<foreach>	Recorre los elementos de un array
<form>	Contenedor para todos los elementos de campo y control
<goto>	Indica la transición a otro elemento o documento
<grammar>	Especifica un reconocimiento de voz o una gramática
<help>	Captura un evento de ayuda
<if>	Cambia el flujo de control mediante una condición
<initial>	Declara inicializaciones lógicas sobre entradas
<item>	Define una opción en una gramática XML
<link>	Implementa un documento o aplicación
<log>	Genera información de depuración de errores
<mark>	Inserta marcadores en los flujos de salida
<media>	Proporciona atributos adicionales al elemento <audio>

<menu>	Define un conjunto de opciones al usuario
<meta>	Define meta información de un documento
<noinput>	Captura un evento de entrada no recibida
<nomatch>	Captura un evento de no reconocimiento de gramática
<one-of>	Construye una gramática con frases alternativas
<option>	Lista opciones simples en lugar de construir una gramática completa
<paragraph>	Especifica el texto a leer en un párrafo
<param>	Envia valores a subdialogos u objetos
<phoneme>	Especifica la pronunción fonética de un texto
<prompt>	Emite al usuario un texto mediante voz
<property>	Configura el comportamiento de la aplicación
<prosody>	Cambia aspectos como el tono, nivel de voz y volumen de una salida
<record>	Graba audio del usuario
<reprompt>	Reproduce de nuevo el mensaje más reciente
<return>	Finaliza un subdiálogo y devuelve el control al diálogo principal
<rule>	Amplia una regla de una gramática XML
<ruleref>	Especifica una regla existente para incluirla en la gramática actual
<say-as>	Informa del tipo del texto y ayuda a renderizarlo
<script>	Especifica un bloque de ECMAScript del lado del cliente
<sentence>	Especifica el texto a leer en una oración
<sub>	Reemplaza el texto por pronunciación
<subdialog>	Invoca un diálogo como un subdiálogo en la sesión activa
<submit>	Envía información
<tag>	Define el valor devuelto de un enunciado del usuario
<throw>	Lanza un evento

<token>	Define palabras que pueden ser dichas por el usuario en una gramática
<transfer>	Provoca que la persona que llama se conecte a otro destino
<value>	Inserta el valor de una variable
<var>	Declara una variable
<vxml>	Declaración inicial que define un documento como una aplicación VoiceXML

Tabla 1: Elementos de VoiceXML

2.3 Plataforma Voxeo Evolution

2.3.1 Introducción

Voxeo Corporation es una de las compañías líderes en el sector de los sistemas de diálogo [VOXEO]. Voxeo ha desarrollado la plataforma VoiceCenter, la cual permite crear, implementar y probar aplicaciones de voz desarrolladas en VoiceXML de forma telefónica y completamente gratuita.

Voxeo es especialista en IVR (*Interactive Voice Response*) y en soluciones de VoIP (*voz sobre IP*) gracias a la infraestructura de servidores y sistemas de desarrollo, apoyados en las tecnologías VoiceXML y CCXML.

Además, Voxeo incluye una web estática local para el alojamiento de las aplicaciones de voz, un número de teléfono dedicado para su uso durante la prueba de las aplicaciones, registro y depuración en tiempo real de la aplicación, conectividad de voz sobre IP, foros de soporte donde se puede interactuar con la Corporación Voxeo y con otros miembros de la comunidad y soporte 24x7.

Por todo lo expuesto anteriormente, se ha elegido la plataforma de desarrollo gratuita Voxeo Evolution para el desarrollo y alojamiento de la inferfaz de voz de nuestra aplicación. La plataforma Voxeo Evolution dispone de todos los módulos necesarios propios de un sistema de diálogo y ofrece un servicio con un número de teléfono local o bien de un número para comunicarnos con la aplicación a través de Skype, para que podamos realizar todas las pruebas necesarias sobre el sistema desarrollado.

2.3.2 Funcionamiento de Voxeo Evolution

Para explicar el funcionamiento de la plataforma de voz Voxeo se va a comparar el modo de operación de un servidor web con el de la red Voxeo.

En la web, los exploradores realizan solicitudes a los servidores web por medio de protocolos de Internet como HTTP. Los servidores web alojan páginas estáticas y aplicaciones dinámicas HTML que se devuelven al explorador. El resultado es una página visual que se ve en un navegador web. Este mecanismo lo podemos observar en la Figura 9.

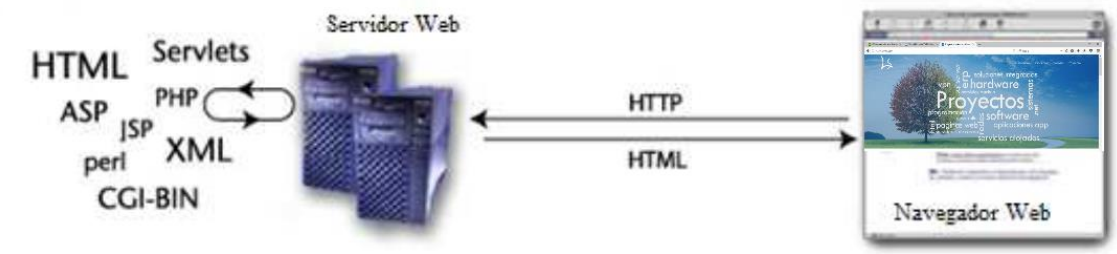

Figura 9. Interpretación de una página web

En la red de Voxeo, las aplicaciones de voz siguen el mismo modelo de navegador que las páginas web estándares. La diferencia es que, en vez de HTML, se utilizan lenguajes de marcado para voz (VoiceXML, CCXML o CallXML) para escribir la página. Cuando estas páginas son procesadas por la red de Voxeo, lo que se devuelve es un audio. La Figura 10 muestra este procedimiento.

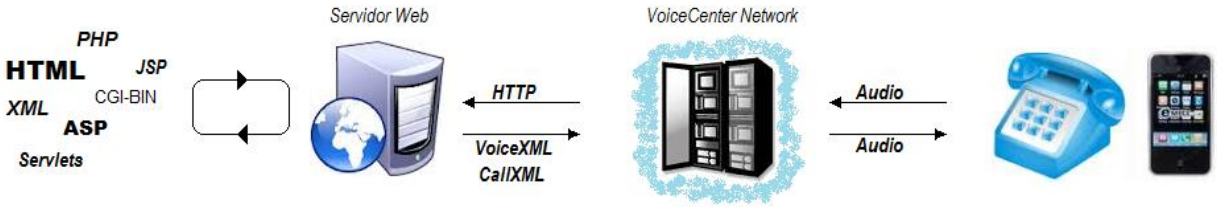

Figura 10. Interpretación de una página VoiceXML

2.3.3 Desarrollo de una aplicación VoiceXML en Voxeo

En este apartado se describe cómo desarrollar y configurar una aplicación VoiceXML usando Voxeo Evolution. Los pasos fundamentales para la utilización de la plataforma Voxeo se detallan a continuación.

El primer paso sería crear una cuenta en la página web y acceder a ella mediante el nombre de usuario y la contraseña. En la Figura 11 se puede ver el aspecto de la interfaz de acceso.

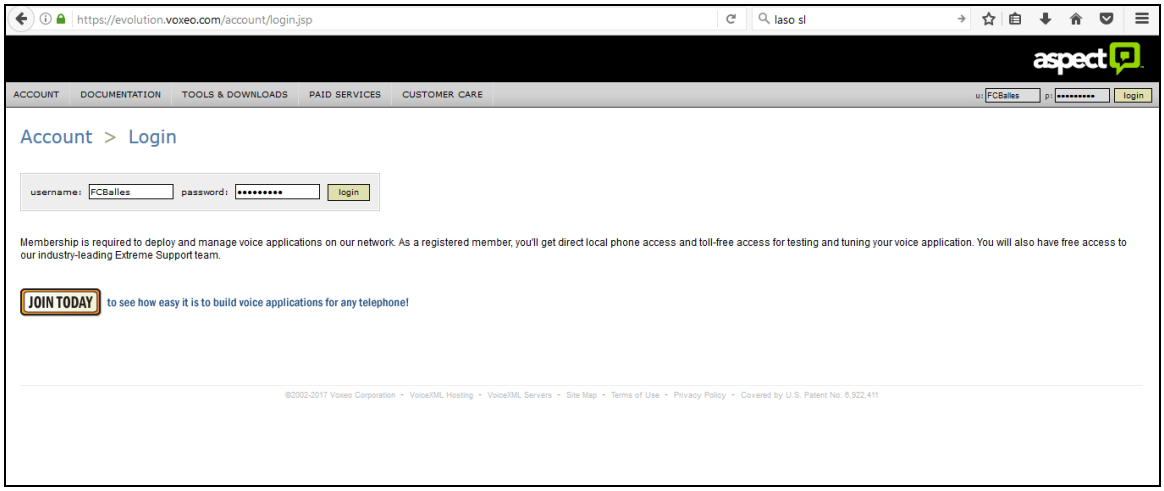

Figura 11. Acceso a la cuenta de Voxeo

Una vez en la página inicial, acceder a la sección de "Application Manager" para crear una nueva aplicación. Este menú de la página inicial se observa en la Figura 12.

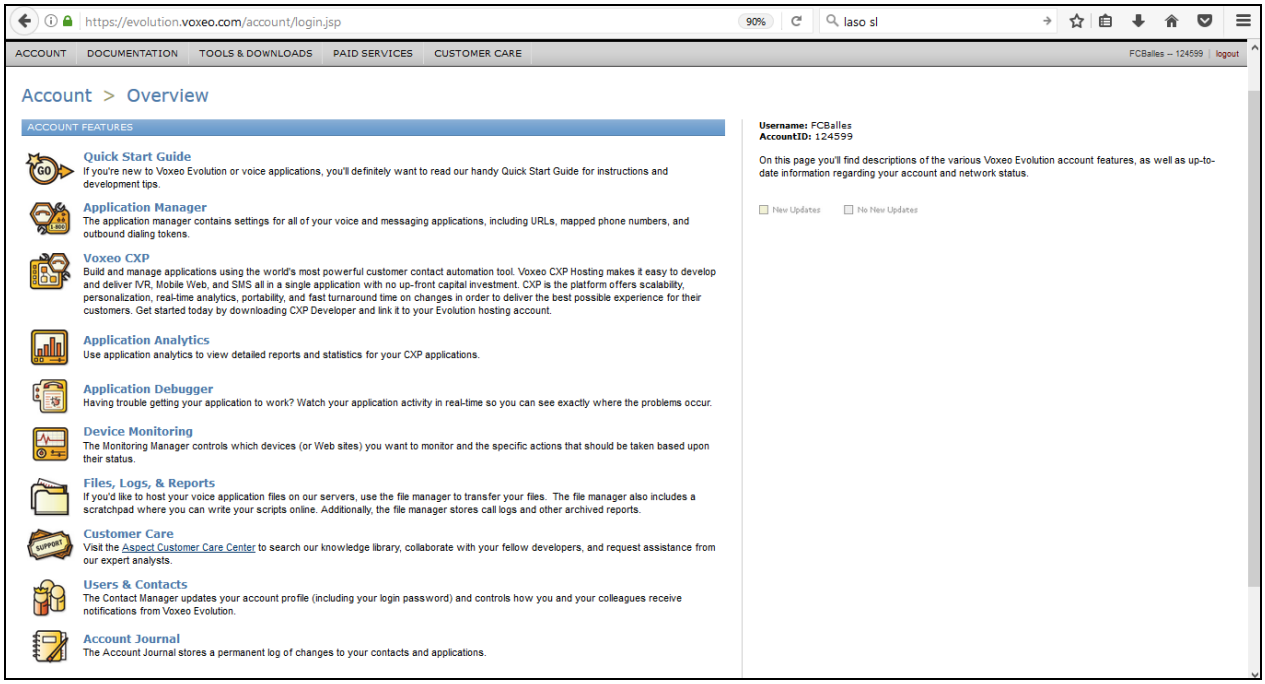

Figura 12. Página principal de la cuenta de Voxeo

Para crear una nueva aplicación hay que indicar el nombre, la página de inicio, la forma de comunicación (llamadas de voz, mensajes de texto o ambos) y el tipo de la aplicación de entre todas las opciones que ofrece. En nuestro caso, el tipo de la aplicación es VoiceXML. La Figura 13 refleja la página que muestra las aplicaciones creadas y desde donde se puede crear una nueva aplicación y en la Figura 14 aparecen las opciones disponibles a la hora de crear una nueva aplicación.

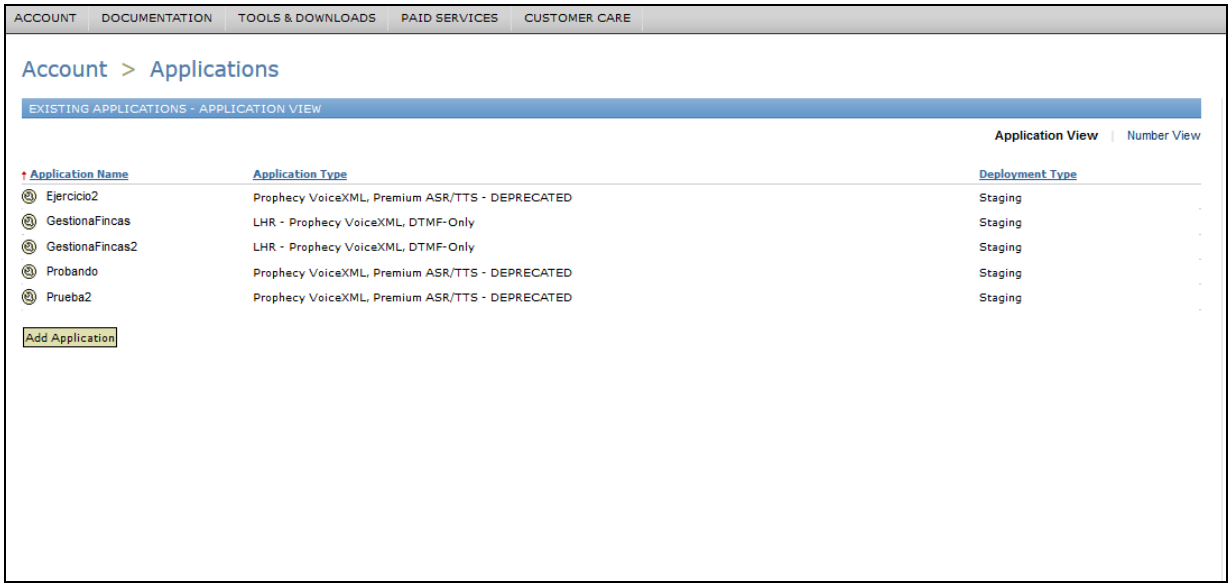

Figura 13. Página principal de la sección "Application Manager"

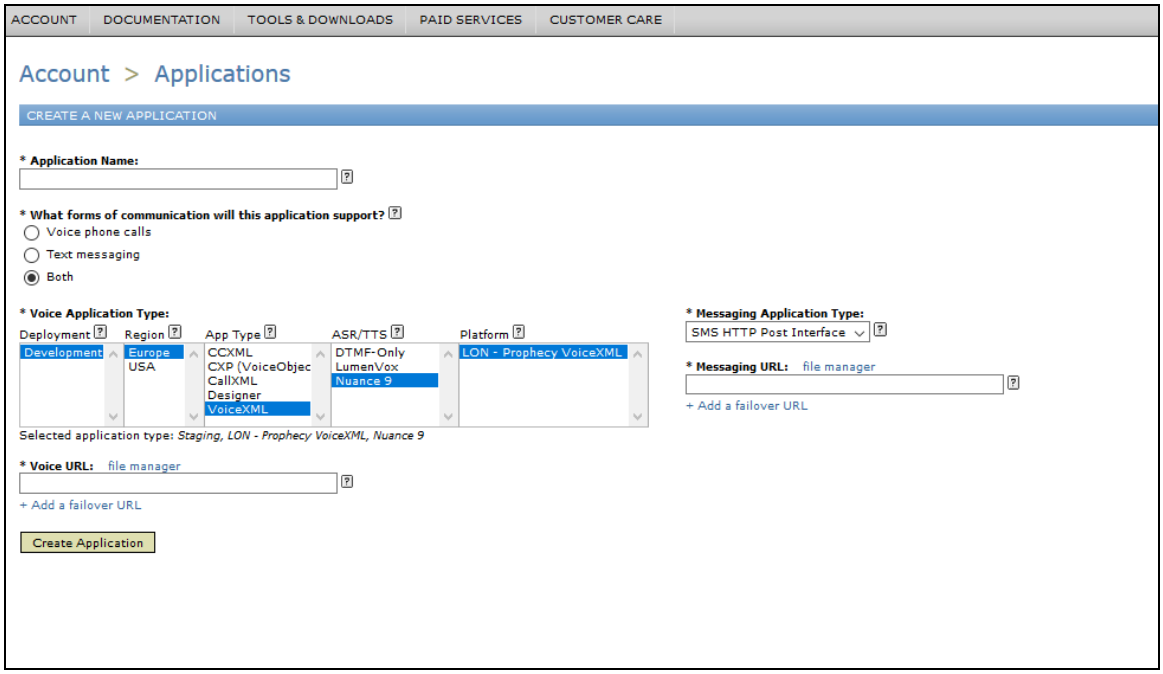

Figura 14. Página de creación de una nueva aplicación.

Una vez se ha creado la aplicación, se puede acceder a ella para modificarla o para conocer los métodos de contacto. En estos métodos se encuentra el número de teléfono asignado al que se debe llamar para ejecutar la aplicación. En la Figura 15 se puede ver la página de información de la aplicación en la que aparecen los datos mencionados anteriomente.

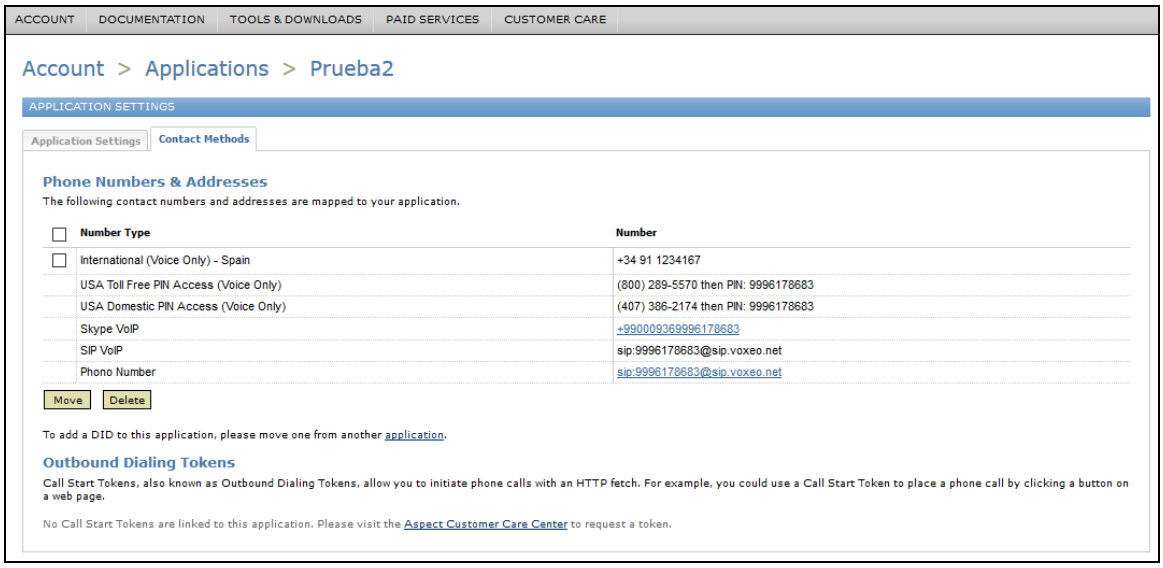

Figura 15. Métodos de contacto de una aplicación.

Conociendo la forma de acceder a la aplicación, solo faltará desarrollar el código VoiceXML y alojarlo en aplicación, o indicar en dicha aplicación la la URL de acceso público con el código a ejecutar.

Si se optara por utilizar el espacio de hospedaje gratuito que ofrece Voxeo para los archivos de la aplicación, éstos habría que subirlos desde la sección "*Files, Logs, Reports*" en el directorio */root/www*. Esta sección se muestra en la Figura 16.

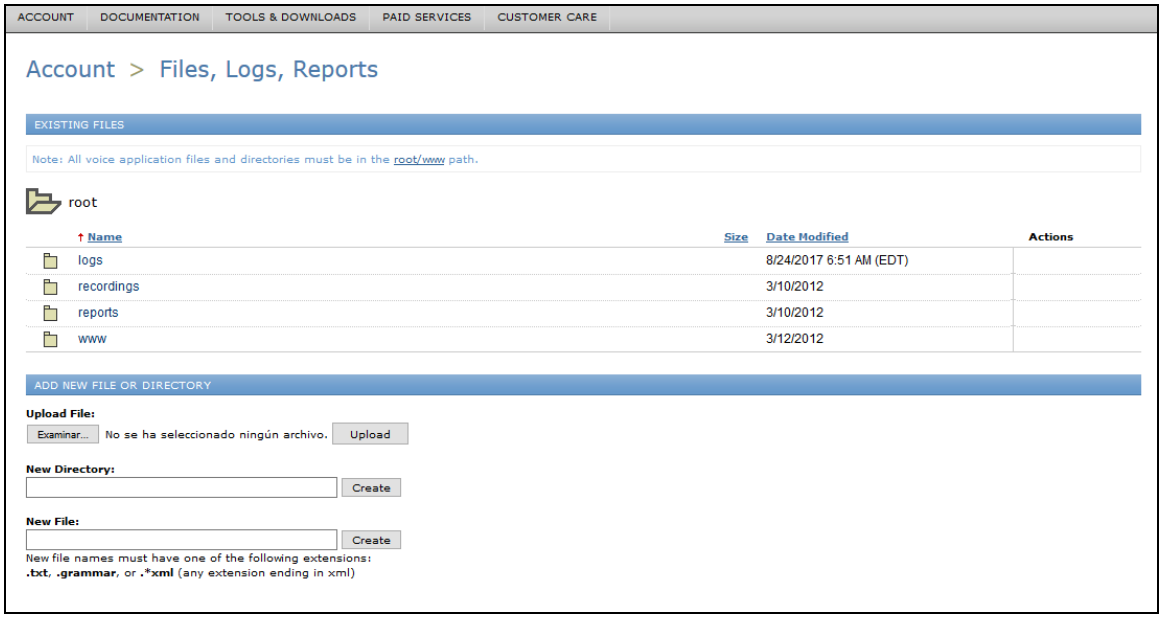

Figura 16. Sección para subir los ficheros de las aplicaciones.

Cuando se suba el código VoiceXML al servidor, ya se podría ejecutar la aplicación. Durante la ejecución de la aplicación, Voxeo también tiene un depurador en la sección "*Application Debugger*". Este depurador muestra la actividad de la aplicación en tiempo real, lo que permite monitorizar su progreso y depurar los errores que se puedan producir. Un ejemplo de lo que ofrece el depurador durante una ejecución se muestra en la Figura 17.

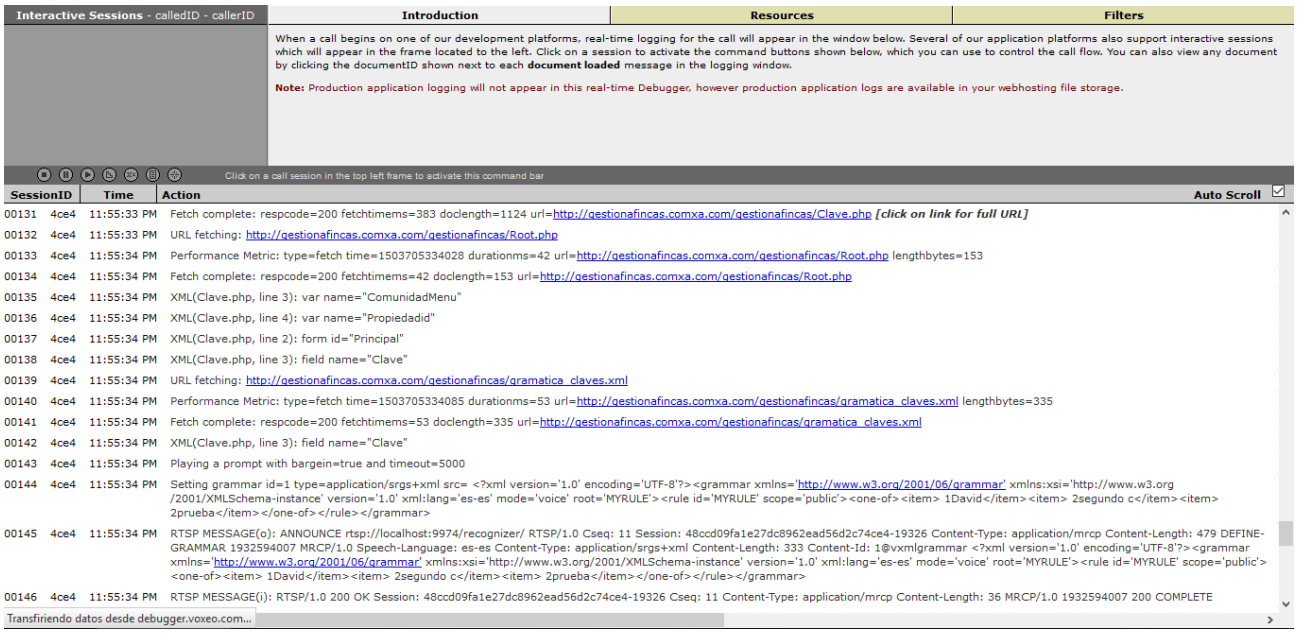

Figura 17. Aspecto del depurador de Voxeo.

Capítulo 3

Descripción general del sistema desarrollado

En este capítulo se hará una presentación del sistema desarrollado explicando en qué consiste y cómo lo lleva a cabo, es decir, se detallará la funcionalidad y la arquitectura del sistema generado.

Además de esta explicación, el capítulo ofrecerá información sobre el servidor web que se ha empleado en el sistema y se analizará detalladamente la base de datos necesaria para este desarrollo, informando sobre las tablas y la información que éstas contienen.

3.1 Funcionalidad y arquitectura

La aplicación desarrollada para este Proyecto Final de Carrera parte de la idea de complementar a los programas de administración de fincas existentes en el mercado. En la empresa Laso S.L. se trabaja diariamente con este tipo de programas, y partiendo de los conocimientos adquiridos, se ha realizado una abstracción de la información con la que trabajan estos programas y la base de datos que requieren. Con esta información se ha generado una nueva base de datos que será con la que interactúe el sistema VoiceXML de este proyecto.

Una de las pantallas del programa de administración de fincas de Laso S.L. aparece en la Figura 18.

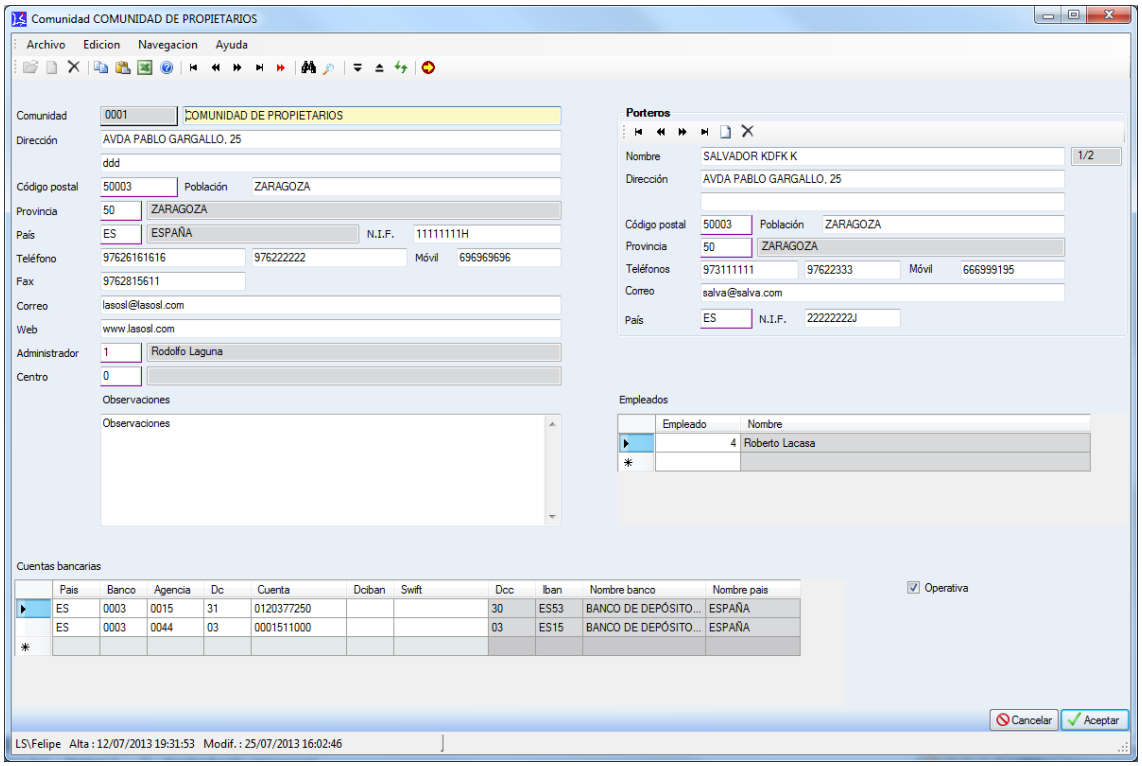

Figura 18. Programa de administración de fincas desarrollado por Laso S.L.

El desarrollo del sistema se ha realizado utilizando una arquitectura cliente-servidor cuyo esquema se puede observar en la Figura 19.

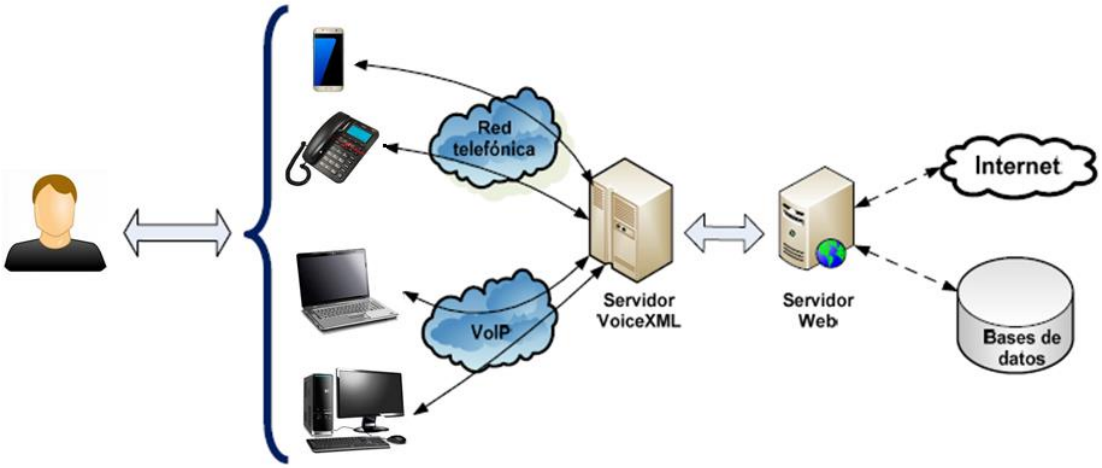

Figura 19. Arquitectura del sistema

La interacción con el sistema comienza cuando el usuario inicia una llamada, bien mediante la línea telefónica tradicional o bien mediante cualquier cliente de VoIP (por ejemplo, Skype).

El servidor VoiceXML obtiene la URL asociada a la aplicación cuyo teléfono ha recibido la llamada. Dicha URL dirige al diálogo VoiceXML inicial del servicio, en el caso de nuestra aplicación, un fichero VoiceXML alojado en el servidor web 000webhost.

El fichero de inicio contendrá el primer diálogo de la aplicación y en función de la interacción con el usuario, redirigirá el diálogo a los formularios correspondientes. Tanto estos formularios como el VoiceXML inicial serán proporcionados mediante peticiones HTTP al servidor web.

Cada uno de los formularios del sistema se encarga de obtener la información necesaria para generar el VoiceXML correspondiente, en función de los parámetros de entrada que usuario ha introducido (mediante la voz o la pulsación de teclas en su teléfono), y accediendo a las tablas de la Base de Datos MySQL en caso de que sea necesario.

El intérprete VoiceXML procesará cada uno de los ficheros VoiceXML por los que transcurra el diálogo, y devolverá la salida al usuario mediante voz sintetizada.

3.2 Servidor Web

El servidor web es el encargado de alojar tanto los ficheros y gramáticas de la aplicación, como la base de datos. Además de este alojamiento, el servidor web se encarga de recibir, en función del flujo de diálogo, la peticiones http del intérprete VoiceXML.

En este proyecto se ha empleado el plan gratuito del servidor web 000webhost [WEBHOST]. La página inicial de una cuenta en este servidor se muestra en la Figura 20.

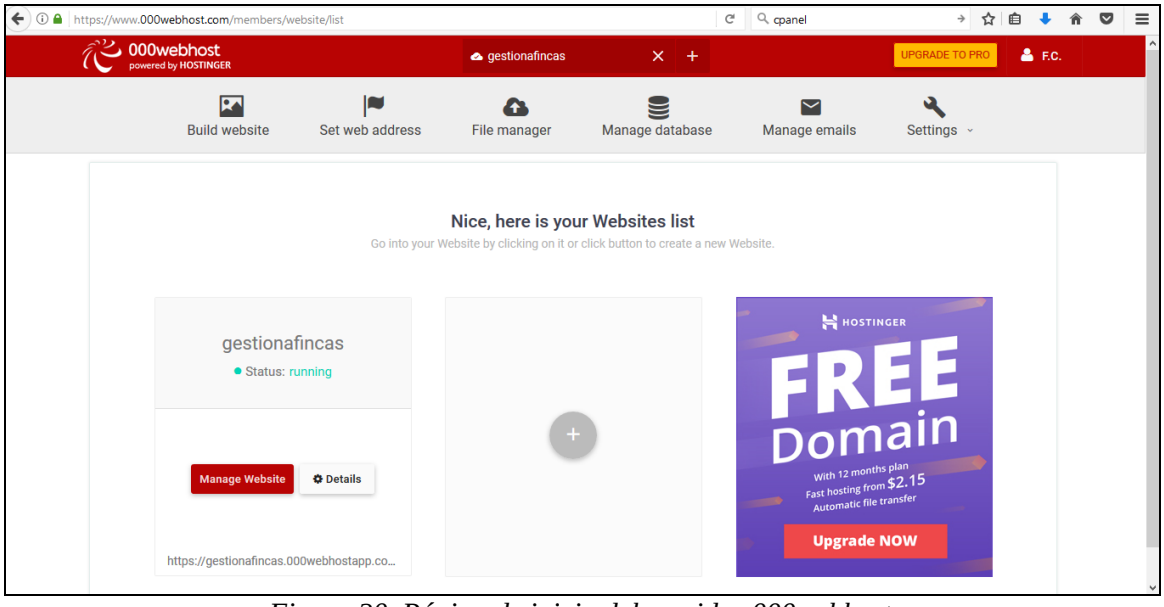

Figura 20. Página de inicio del servidor 000webhost

Las características que ofrece este servidor web son las siguientes:

- 1 GB de espacio de almacenamiento.
- 100 GB de ancho de banda.
- 2 sitios web.
- 2 bases de datos MySQL.
- 5 redirecciones de correo electrónico.
- Ultima versión de CPanel con PHP y MySQL.
- Auto instalador de scripts.
- 99 % online garantizado.

3.3 Base de datos

El servidor web comentado en el apartado anterior permite la utilización de una base de datos MySQL, y es ésta la base de datos que hemos empleado en nuestro sistema.

MySQL es la base de datos de código abierto más popular del mercado. Gracias a su rendimiento probado, a su fiabilidad y a su facilidad de uso, MySQL se ha convertido en la base de datos líder elegida para las aplicaciones basadas en web y utilizada por propiedades web de un alto perfil. [ORACLE]

Además, el servidor web también ofrece phpMyAdmin, una aplicación que permite realizar el diseño de la base de datos de manera visual, para posteriormente generar el código de la base de datos de forma automática. Esto permite una mayor flexibilidad y rapidez en el momento de realizar modificaciones sobre el diseño original durante la creación de la aplicación.

La versión de phpMyAdmin utilizada es la 4.7.4. cuya interfaz inicial se muestra en la Figura 21.

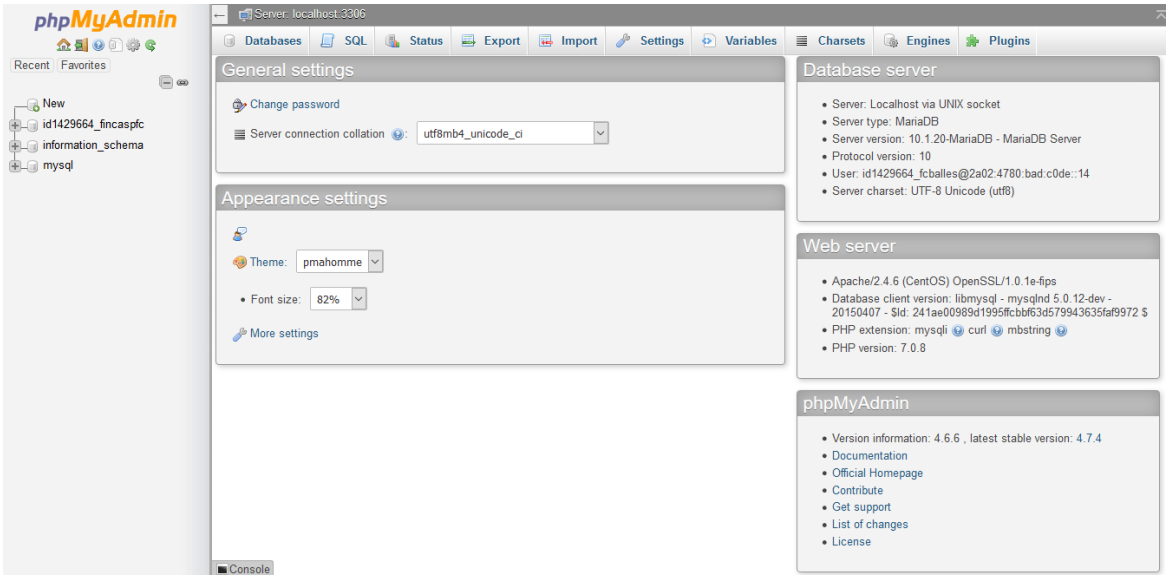

Figura 21. Pantalla inicial del administrador de base de datos phpMyAdmin

Para el almacenamiento de toda la información necesaria para la aplicación se ha utilizado una única base de datos formada por un total de 15 tablas, cuyo diseño se muestra en la Figura 22.

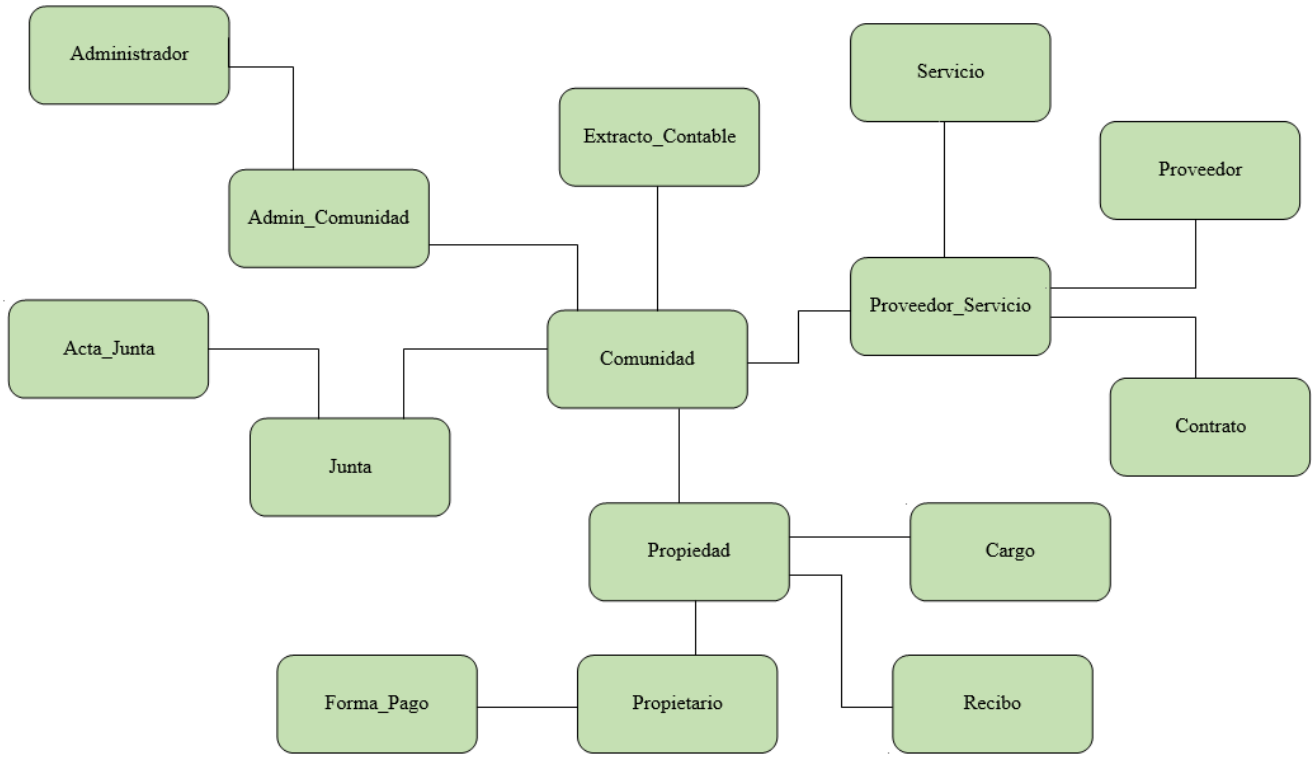

Figura 22. Relaciones de las tablas de la base de datos

La descripción de las tablas de la base de datos y su contenido se detalla a continuación.

Tabla Comunidad

Es la tabla principal que almacena la información de todas las comunidades gestionadas por nuestro sistema de administración de fincas. Sus atributos se detallan en la Tabla 2.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador de la tabla
Nombre	varchar(100)	Nombre de la comunidad
Dirección	varchar(100)	Dirección de la comunidad
Localidad	varchar(100)	Nombre de la localidad
Dimensiones	double	Tamaño de la comunidad en metros cuadrados

Tabla 2: Atributos de la tabla Comunidad

Tabla Administrador

Almacena la información de los administradores de fincas que pueden utilizar el sistema. Sus atributos se detallan en la Tabla 3.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador del administrador
Nombre	varchar(60)	Nombre del administrador
Clave	varchar(60)	Contraseña encriptada de acceso al sistema
Correo	varchar(100)	Correo electrónico

Tabla 3: Atributos de la tabla Administrador

Tabla Admin_Comunidad

Es la tabla encargada de guardar las relaciones de los administradores y las comunidades que administran. Un mismo administrador puede administrar varias comunidades, al igual que una comunidad puede ser administrada por varios administradores. Sus atributos se detallan en la Tabla 4.

ATRIBUTO	TIPO	DESCRIPCIÓN
Comunidadid	int(11)	Referencia la clave de la comunidad
Administradorid	Int(11)	Referencia la clave del administrador

Tabla 4: Atributos de la tabla Admin_Comunidad

Tabla Propiedad

Almacena la información de las propiedades que formar parte de las comunidades. Sus atributos se detallan en la Tabla 5.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador de la propiedad
Nombre	varchar(100)	Nombre de la propiedad
Propietarioid	int(11)	Referencia de la clave del propietario
Inquilino	varchar(100)	Nombre del inquilino en el caso de que la propiedad esté arrendada
Coeficiente	Doublé	Coeficiente de reparto
Dimensiones	Doublé	Tamaño de la propiedad en metros cuadrados
Comunidadid	int(11)	Referencia a la comunidad a la que pertenece
Clave	varchar(60)	Contraseña encriptada con la que el propietario accede al sistema

Tabla 5: Atributos de la tabla Propiedad

Tabla Propietario

Almacena la información de los propietarios, dueños de cada propiedad. Existirá un propietario para cada propiedad. Sus atributos se detallan en la Tabla 6.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador del propietario
Nombre	varchar(100)	Nombre del propietario
Nif	varchar(15)	NIF del propietario
NumCuenta	varchar(40)	Numero de cuenta registrada para realizar los pagos de la comunidad
FormaPagoId	int(11)	Referencia al tipo de pago seleccionado
Correo	varchar(100)	Correo electrónico

Tabla 6: Atributos de la tabla Propietario

Tabla Forma_Pago

Identifica todos los tipos de formas de pago que tienen los propietarios. Sus atributos se detallan en la Tabla 7.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador de la forma de pago
Nombre	varchar(100)	Nombre de la forma de pago

Tabla 7: Atributos de la tabla Forma_Pago

Tabla Recibo

Almacena la información de recibos que las propiedades pagan a la comunidad. Sus atributos se detallan en la Tabla 8.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador del recibo
Propiedadid	int(11)	Referencia de la propiedad a la que pertenece este recibo
Importe	float	Importe del recibo
Pagado	tinyint(1)	Indicador que muestra si el recibo está pagado

Tabla 8: Atributos de la tabla Recibo

Tabla Cargo

Almacena los cargos que tiene cada comunidad, así como la propiedad que lo representa. Sus atributos se detallan en la Tabla 9.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador del recibo
Propiedadid	int(11)	Referencia de la propiedad que lleva a cabo el cargo
Comunidadid	int(11)	Referencia a la comunidad que tiene el cargo
Nombre	varchar(100)	Nombre del cargo

Tabla 9: Atributos de la tabla Recibo

Tabla Junta

Almacena las reuniones o juntas vecnales que se celebran en la comunidad. Sus atributos se detallan en la Tabla 10.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador de la junta vecinal
Comunidadid	int(11)	Referencia de la comunidad que celebra la junta
Fecha	Date	Fecha en la que se realiza la junta
Extraordinaria	tinyint(1)	Indicador que muestra si la junta es extraordinaria o no

Tabla 10: Atributos de la tabla Junta

Tabla Acta_Junta

Almacena el contenido de las reuniones o juntas vecinales que se celebran en la comunidad. Sus atributos se detallan en la Tabla 11.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador del acta de la junta
Juntadid	int(11)	Referencia de la junta celebrada
Orden_dia	varchar(20000)	Puntos del orden del día que se tratan en la junta vecinal
Contenido	varchar(40000)	Resultado de todo aquello que se ha tratado en la junta

Tabla 11: Atributos de la tabla Acta_Junta

Tabla Extracto_Contable

Almacena los movimientos contables que se realizan en comunidad. Sus atributos se detallan en la Tabla 12.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	bigint(20)	Identificador del movimiento contable
Comunidadid	int(11)	Referencia de la comunidad

Fecha	datetime	Fecha en la que se realizó el movimiento contable
Importe	float	Importe del movimiento contable
Debe	tinyint(1)	Indicador que muestra si el movimiento contable es para sacar dinero o para meterlo
Concepto	varchar(600)	Concepto del movimiento contable

Tabla 12: Atributos de la tabla Extracto_Contable

Tabla Proveedor

Almacena información de los proveedores que pueden ser tenidos en cuenta para la realización de cualquier servicio necesario para la comunidad. Sus atributos se detallan en la Tabla 13.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador del proveedor
Nombre	varchar(100)	Nombre del proveedor
Telefono	varchar(20)	Teléfono de contacto
NIF	varchar(15)	NIF del proveedor

Tabla 13: Atributos de la tabla Proveedor

Tabla Servicio

Identifica todos los tipos de servicios que pueden llevarse a cabo en una comunidad. Sus atributos se detallan en la Tabla 14.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador del servicio
Nombre	varchar(100)	Nombre del servicio

Tabla 14: Atributos de la tabla Servicio

Tabla Proveedor_Servicio

Es la tabla encargada de guardar las relaciones de los proveedores con los servicios que ofrece para cada comunidad. Un mismo proveedor puede ofrecer varios servicios a varias comunidades, del mismo modo que un servicio puede ser ofrecido por varios proveedores diferentes. Sus atributos se detallan en la Tabla 15.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador de la relación
Comunidadid	int(11)	Referencia de la comunidad
Proveedorid	int(11)	Referencia del proveedor que ofrece el servicio
Servicioid	int(11)	Referencia del servicio ofrecido

Tabla 15: Atributos de la tabla Proveedor_Servicio

Tabla Contrato

Almacena los contratos que la comunidad tiene con los distintos proveedores. Sus atributos se detallan en la Tabla 16.

ATRIBUTO	TIPO	DESCRIPCIÓN
Id	int(11)	Identificador del contrato
Gestionid	int(11)	Referencia de la gestión (relación del proveedor y servicio para una comunidad) que cumple el contrato
Fecha	date	Fecha de inicio del contrato
Precio	float	Importe del contrato
FechaFin	date	Fecha de finalización del contrato

Tabla 16: Atributos de la tabla Contrato

Capítulo 4

Explicación detallada del sistema desarrollado

En este capítulo se analiza en detalle el sistema ofreciendo una descripción del mismo y de cada uno de los módulos en los que se puede dividir. Para la descripción de cada uno de estos módulos del sistema se incluirá una explicación de su funcionamiento y su arquitectura, así como el detalle de algún escenario de uso.

4.1 Descripción general

Como se ha comentado en apartados anteriores, el sistema desarrollado en este Proyecto Final de Carrera consiste en ofrecer una nueva funcionalidad a los sistemas de administración de fincas. Para la ejecución de este sistema será necesario acceder mediante una llamada de teléfono al número asociado.

El acceso a la interfaz de voz del sistema se realizará llamando desde cualquier teléfono al número +34 91 1234171, o bien llamar gratuitamente a través de una cuenta de Skype al número +990009369996113489.

La interfaz de voz del sistema se encuentra alojada en el servidor 000webhost. En este servidor se encuentran todos los ficheros necesarios, entre los cuales está el fichero inicial *Inicio.php* que entre en ejecución con la llamada comentada anteriormente. La

dirección concreta de este fichero inicial es http://gestionafincas.comxa.com/gestionafincas/Inicio.php

Para poder comunicarse con el sistema de forma oral, éste requerirá de muchas gramáticas durante la sesión. Estas gramáticas se irán generando de forma dinámica, mediante consultas a la base de datos, a medida que se vayan necesitando. La configuración de acceso a la base de datos se muestra en la Figura 23.

```
$DBHost="localhost";
$DBUser="id1429664_fcballes";
$DBPass="uc3m000111";
$db = "id1429664_fincaspfc";

$conexion=mysql_connect($DBHost, $DBUser, $DBPass)or die(mysql_error());
if (!$conexion)
{
    echo "Error conectando a la base de datos.";
    exit();
}

$resultado=mysql_select_db($db,$conexion);
if (!$resultado)
{
    echo "Error seleccionando la base de datos.";
    exit();
}
```

Figura 23. Código para la conexión a la BBDD

La ejecución del sistema comienza con un mensaje de bienvenida y la solicitud de una contraseña para la comunidad que se desea gestionar. Con esta identificación, el sistema distingue entre dos tipos de usuarios: administradores y administrados.

Dependiendo del tipo de usuario, el sistema ofrecerá un menú u otro con diferentes opciones para trabajar con la comunidad. En función de la opción del menú seleccionada, el sistema ejecutará el módulo correspondiente a cada opción. Un esquema de la organización de estos módulos del sistema se encuentra en la Figura 24.

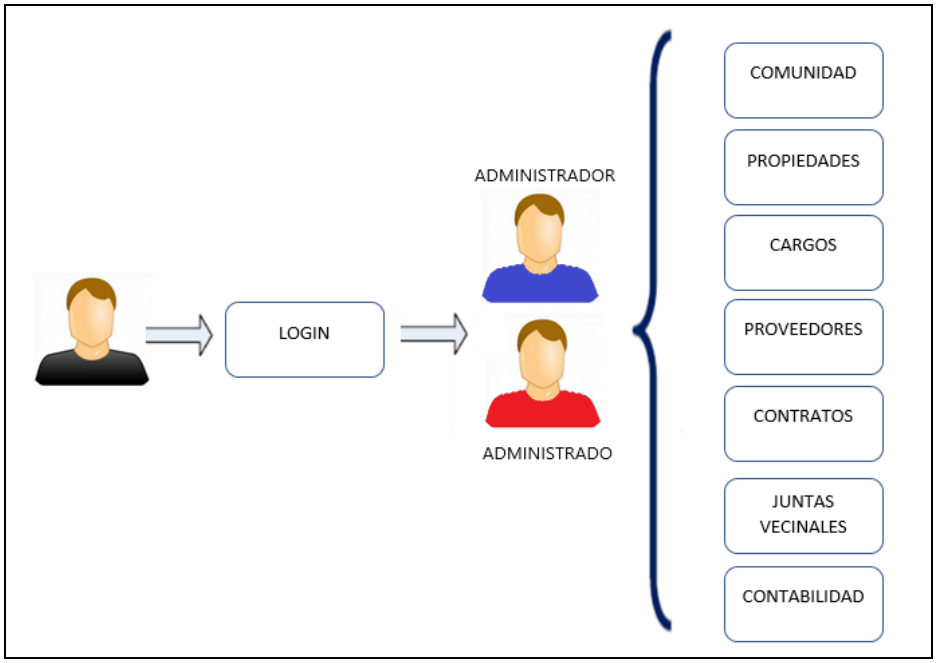

Figura 24. Organización de los módulos del sistema

4.2 Módulos del sistema

4.2.1 Login

El módulo de login implementa el primer diálogo que se le proporciona al usuario al llamar al número de teléfono del portal de voz.

En este primer proceso, el sistema comenzará con un mensaje de bienvenida en el que se solicitará el nombre de la comunidad con la que se quiere trabajar. Una vez se conozca la comunidad, se solicitará también una contraseña a través de la cual, el sistema será capaz de diferenciar el tipo de usuario que está accediendo y le ofrecerá las opciones indicadas para su perfil. Dependiendo de la opción seleccionada, se llamará a un módulo u otro.

Este es el único módulo del sistema que es independiente del tipo de usuario. Es a partir del login cuando la ejecución de cada tipo de usuario irá por caminos diferentes. Los dos tipos de usuarios tendrán la opción de ejecutar los mismos siete módulos restantes, sin embargo, las clases ejecutadas serán siempre diferentes y en algunos casos, la información y las opciones disponibles, también.

ESTRUCTURA Y FUNCIONAMIENTO

El fichero inicial de este proceso es *Inicio.php*. Este fichero ofrece al usuario un mensaje de bienvenida y solicita una comunidad para trabajar. Para poder comprender la comunidad seleccionada, llamará a la clase *GestionGramatica.php*. Esta clase es la encargada de conectar con la base de datos y generar todas las gramáticas que sean necesarias. En este caso generará la gramática *gramatica_comunidades.xml* que contendrá el nombre de todas las comunidades dadas de alta en el sistema y cuyo ejemplo se muestra en la Figura 25.

```
<?xml version= "1.0" encoding="UTF-8"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="es-ES" version="1.0" root="MYRULE">
<rule id="MYRULE">
<one-of>
<item>Jardines de Vistabella<tag>"Jardines de Vistabella";</tag> </item>
<item>Burgos<tag>"Burgos";</tag> </item>
<item>Luces de Bohemia<tag>"Luces de Bohemia";</tag> </item>
<item>Terrazas de Miralbueno<tag>"Terrazas de Miralbueno";</tag> </item>
<item>Fuentenueva<tag>"Fuentenueva";</tag> </item>
</one-of>
</rule>
</grammar>/
```

Figura 25. Ejemplo de gramatica_comunidades.xml

Tanto en el proceso de reconocmiento de esta gramática, como en todos lo demás procesos del sistema, se ha creado un evento que solicitará la repetición de lo que el usuario dice, en caso de que la gramática no lo entienda, y otro evento en caso de que no se diga nada o no se haya sido capaz de escuchar nada.

Cuando se haya seleccionado una comunidad con la que trabajar, entrará en ejecución la clase *Clave.php*, que solicitará una clave de acceso y generará una gramática con las claves de los administradores y administrados que haya en la comunidad seleccionada (*gramatica_claves.xml*). Cabe destacar que las claves se encuentran encriptadas mediante base64 en la base de datos, por lo que para generar la gramática es necesario desencriptar esta información gracias a una clave que conocemos de antemano. Un ejemplo de *gramática_claves.xml* se muestra en la Figura 26.

```
<?xml version= "1.0" encoding="UTF-8"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="es-ES" version="1.0" root="MYRULE">
<rule id="MYRULE">
<one-of>
<item>1David<tag>"1David";</tag> </item>
<item>2segundo c<tag>"2segundo c";</tag> </item>
<item>2prueba<tag>"2prueba";</tag> </item>
</one-of>
</rule>
</grammar>
```

Figura 26. Ejemplo de gramatica_claves.xml

Como se puede observar en la anterior imagen, las claves generadas tienen un número, 1 o 2, antes de la propia clave. Este número es el que indica el tipo de usuario, siendo el 1 para los administradores y el 2 para los administrados. Este número no se tiene en cuenta a la hora de analizar la clave que dice el usuario y compararlo con la gramática existente.

Una vez se haya obtenido la clave pueden darse dos situaciones:

- Si el usuario es administrador, se llamará a la clase *MenuAdmin.php* y ésta ofrecerá al usuario todas las opciones que puede consultar: descripción, propiedades, cargos, proveedores, contratos, juntas vecinales y contabilidad.

- Si el usuario es administrado, se llamará a la clase *ObtienePropiedad.php* y ésta a su vez volverá a llamar a *GestiónGramática.php* para obtener el identificador de propiedad a la que pertenece en usuario. Una vez identificada la propiedad se ejecutará la clase *MenuUser.php* que al igual que en el caso anterior, ofrecerá al usuario todas las opciones que puede consultar: información de la comunidad, información de la propiedad, cargos, proveedores, contratos, juntas vecinales y contabilidad.

El diagrama de secuencia de este módulo de Login se muestra en Figura 27.

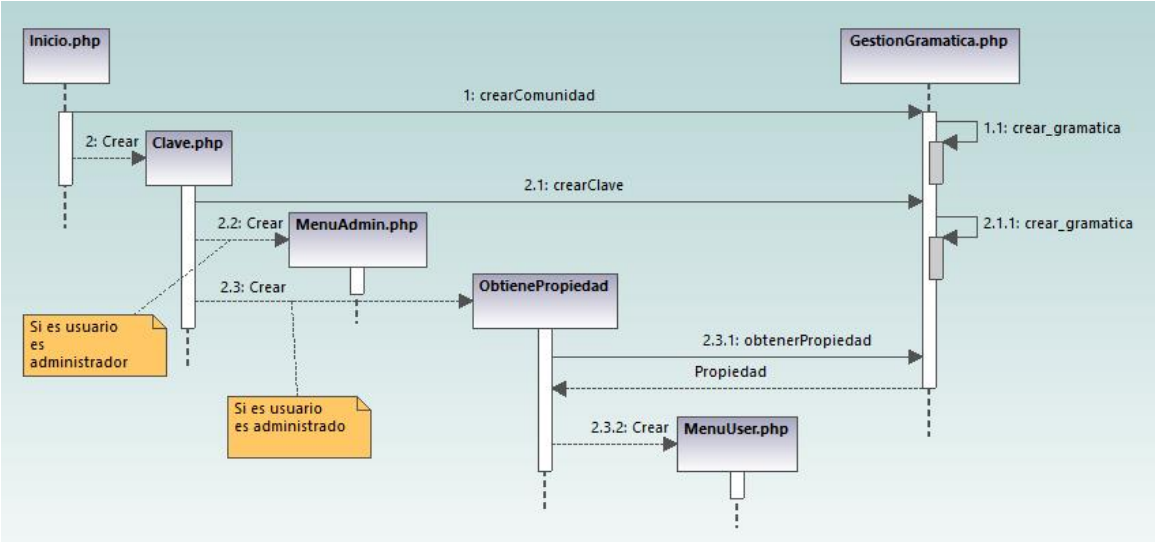

Figura 27. Diagrama de secuencia del módulo Login

ESCENARIOS DE USO

A continuación, en las Figuras 28 y 29, se mostrarán algunos diálogos que ejemplifican el funcionamiento de este módulo siendo "S" el sistema y "U" el usuario.

S: Bienvenido a la aplicación de gestión de fincas. A continuación nombre la comunidad que desea gestionar.

U: Burgos.

S: Ahora diga la clave que tiene en la comunidad.

U: David.

S: Bienvenido al menú de Administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 28. Escenario del módulo Login para administrador

S: Bienvenido a la aplicación de gestión de fincas. A continuación nombre la comunidad que desea gestionar.

U: Burgos.

S: Ahora diga la clave que tiene en la comunidad.

U: Prueba.

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 29. Escenario del módulo Login para administrado

4.2.2 Comunidad

El módulo de comunidad implementa la primera de las opciones que el sistema ofrece a los usuarios y consiste en una consulta de la información general de la comunidad.

Los dos tipos de usuarios del sistema, administradores y administrados, tienen diferente menú de opciones y la secuencia de ejecución del sistema es diferente para cada uno de ellos, sin embargo, la información obtenida en este módulo será la misma independientemente del tipo de usuario (cada comunidad tiene su información propia, pero la estructura de la información es siempre la misma)

ESTRUCTURA Y FUNCIONAMIENTO

Como se ha comentado anteriormente, la información de este módulo no varía dependiendo del tipo de usuario, sin embargo, las clases que entran en acción sí que son diferentes para cada tipo de usuario.

- La ejecución del usuario administrador partirá de las opciones ofrecidas por el menú de la clase *MenuAdmin.php*. Al seleccionar la primera opción, "descripción", se solicitará la descripción de la comunidad a la clase *GestionGramatica.php*, ésta devolverá la información requerida y la clase *MenuAdmin.php* se encargará de mostrar el texto para que el interprete de voz lo lea.

Una vez se ha leído la información, se llamará a la clase *DescripcionOpciones.php* que le indicará al usuario un nuevo menú con dos opciones para llevar a cabo, repetir la descripción o volver al menú principal.

En caso de repetir descripción será la clase *DescripcionOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la descripción se vuelve a preguntar por las dos opciones a realizar.

En el caso de volver al menú principal, se volverá a llamar a la clase *MenuAdmin.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Comunidad para el usuario administrador se muestra en Figura 30.

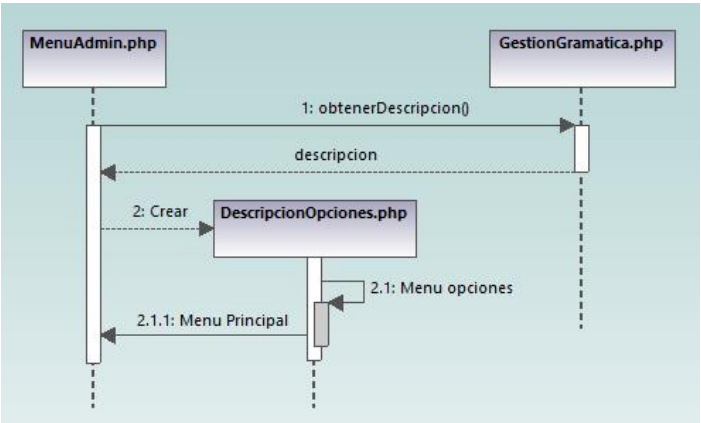

Figura 30. Diagrama de secuencia del módulo Comunidad (administrador)

- La ejecución del usuario administrado será muy similar a la explicada anterioremente para el usuario administrador. La única diferencia son las clases que se ejecutan, y es que el menú del administrado se encuentra en la clase *MenuUser.php*, y la clase que se ejecuta tras la obtención de la descripción es la clase *DescripciónUsuarioOpciones.php.*

El diagrama de secuencia de este módulo de Comunidad para el usuario administrado se muestra en Figura 31.

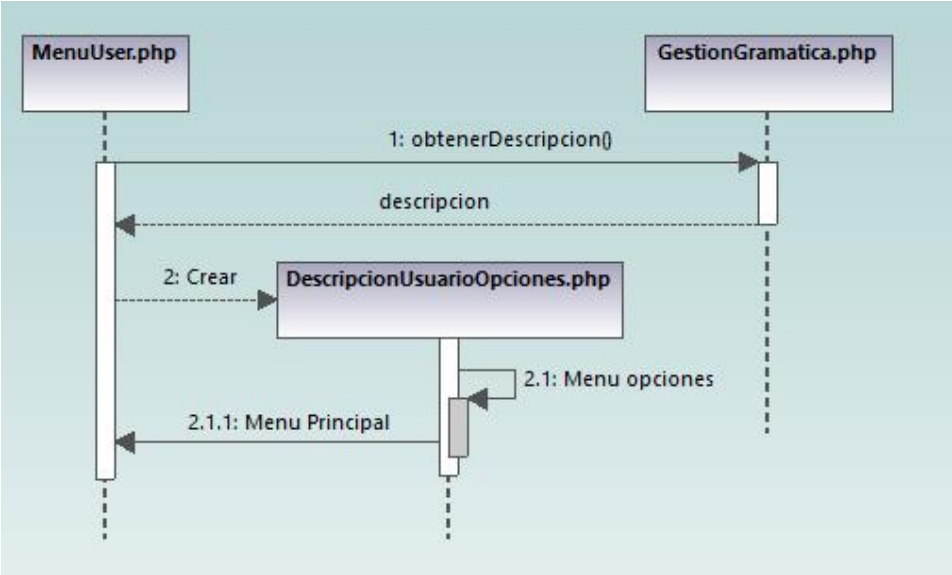

Figura 31. Diagrama de secuencia del módulo Comunidad (administrado)

ESCENARIOS DE USO

En esta ocasión, en las Figuras 32 y 33 se muestra un escenario para cada tipo de usuario.

S:	Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.
U:	Descripción.
S:	La comunidad Burgos se encuentra en la calle Burgos y tiene una dimensión de 800 metros cuadrados divididos en 5 propiedades.
S:	Que desea hacer ahora: repetir descripción o volver al menú principal.
U:	Volver al menú principal.
S:	Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 32. Escenario de uso del módulo Comunidad (administrador)

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

U: Información de la comunidad

S: La comunidad Burgos se encuentra en la calle Burgos y tiene una dimensión de 800 metros cuadrados divididos en 5 propiedades

S: Que desea hacer ahora: repetir descripción o volver al menú principal

U: Repetir descripción

S: La comunidad Burgos se encuentra en la calle Burgos y tiene una dimensión de 800 metros cuadrados divididos en 5 propiedades

S: Que desea hacer ahora: repetir descripción o volver al menú principal

U: Volver al menú principal

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 33. Escenario de uso del módulo Comunidad (administrado)

4.2.3 Propiedades

El módulo de propiedades implementa la segunda de las opciones que el sistema ofrece a los usuarios y consiste en una consulta de la información de las propiedades.

En este caso, el usuario administrado solo tendrá la posibilidad de consultar la información de su propiedad, sin embargo, el administrador podrá consultar la información de cualquiera de las propiedades que pertenezcan a la comunidad

ESTRUCTURA Y FUNCIONAMIENTO

En este módulo el comportamiento del sistema comienza a tener diferencias significativas dependiendo del tipo de usuario.

- La ejecución del usuario administrador partirá de las opciones ofrecidas por el menú de la clase *MenuAdmin.php*. Al seleccionar la segunda opción, "propiedades", se llamará a la clase *InformacionGlobal.php,* esta clase preguntará al administrador el piso que quiere consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada "*gramatica_pisos.xml"* capaz de comprender este piso. Un ejemplo de esta gramática se muestra en la Figura 34.

```
<?xml version= "1.0" encoding="UTF-8"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="es-ES" version="1.0" root="MYRULE">
<rule id="MYRULE">
<one-of>
<item>local<tag> out.Propietario="El propietario del local es Felipe. No tiene inquilinos y su coeficiente de reparto es 5.5
para unas dimensiones de 110 metros cuadrados";</tag> </item>
<item>PrimeroA<tag> out.Propietario="El propietario del PrimeroA es Alberto. Su inquilino es Enrique Pérez y su coeficiente
de reparto es 2 para unas dimensiones de 200 metros cuadrados";</tag> </item>
</one-of>
</rule>
</grammar>
```

Figura 34. Ejemplo de gramatica_pisos.xml

Cuando se haya obtenido el piso a consultar, la gramática generada, cuyo ejemplo se muestra en la imagen anterior, tendrá asociada una descripción para este piso, y la clase *InformacionGlobal.php* la mostrará para que el interprete de voz la lea.

Tras leer la información del piso, se ejecutará la clase *InformacionOpciones.php*, y ésta, preguntará al administrador que qué desea hacer, ofreciendo un menú con tres opciones: repetir información, consultar otro piso o volver al menú principal.

En caso de repetir información será la clase *InformacionOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las tres opciones a realizar.

Si se decide consultar otro piso, se volverá a ejecutar la clase *InformacionGlobal.php*, que volverá a preguntar por el piso a consultar y repetirá de nuevo todo el proceso.

Finalmente, en el caso de volver al menú principal, se volverá a llamar a la clase *MenuAdmin.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Propiedades para el usuario administrador se muestra en Figura 35.

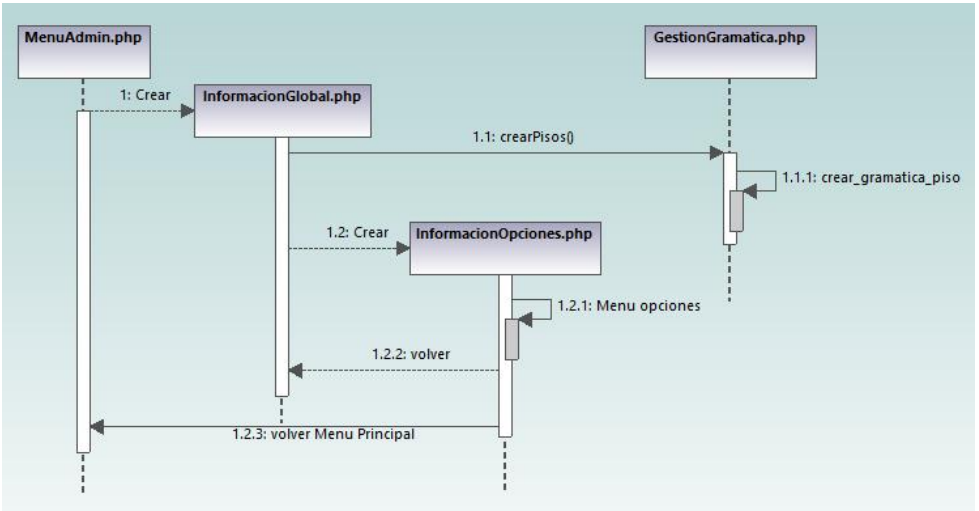

Figura 35. Diagrama de secuencia del módulo Propiedades (administrador)

- El usuario administrado solo puede consultar su propiedad, por lo tanto, la secuencia de ejecución es bastante más simple que la del administrador.

Partiendo de las opciones ofrecidas por el menú de la clase *MenuUser.php*, al seleccionar la segunda opción, "información de la propiedad", se solicitará la descripción de la propiedad del usuario a la clase *GestionGramatica.php*, ésta devolverá la información requerida y la clase *MenuUser.php* se encargará de mostrar el texto para que el interprete de voz lo lea.

Una vez se ha leído la información, se llamará a la clase *InformacionPisoOpciones.php* que le indicará al usuario un nuevo menú con 2 opciones para llevar a cabo, repetir la información o volver al menú principal.

En caso de repetir información será la clase *InformacionPisoOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información, se vuelve a preguntar por las dos opciones a realizar.

En el caso de volver al menú principal, se volverá a llamar a la clase *MenuUser.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Propiedades para el usuario administrado se muestra en Figura 36.

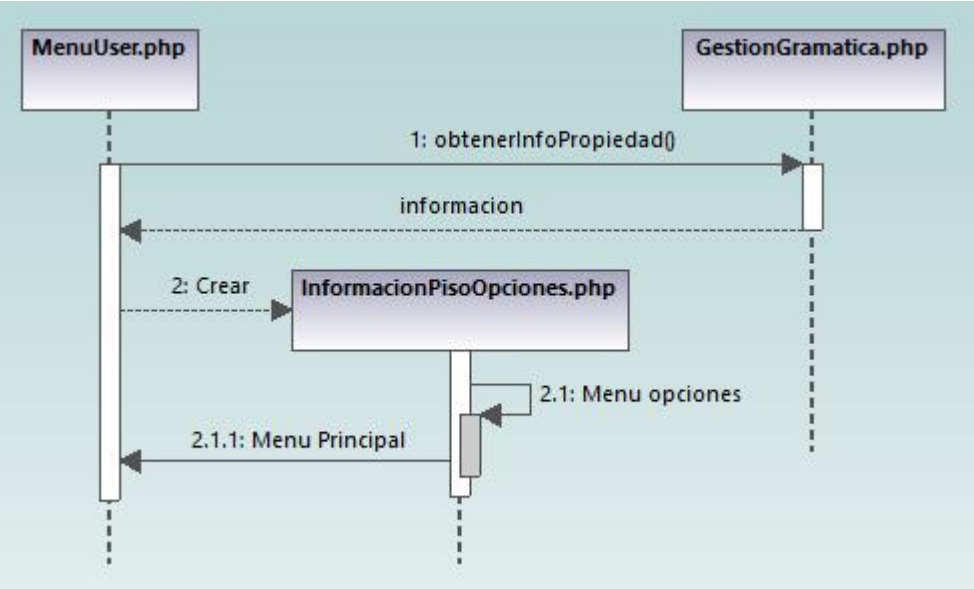

Figura 36. Diagrama de secuencia del módulo Propiedades (administrado)

ESCENARIOS DE USO

En las Figuras 37 y 38 se muestra un escenario para cada tipo de usuario.

S: Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

U: Propiedades.

S: Que piso desea consultar?

U: Local.

S: El propietario del local es Felipe. No tiene inquilinos y su coeficiente de reparto es 5.5 para unas dimensiones de 110 metros cuadrados

S: Que desea hacer ahora: repetir información, consultar otro piso o volver al menú principal.

U: Consultar otro piso.

S: Que piso desea consultar?

U: Primero A.

S: El propietario del local es Alberto. Su inquilino es Enrique Pérez y su coeficiente de reparto es 2 para unas dimensiones de 200 metros cuadrados

S: Que desea hacer ahora: repetir información, consultar otro piso o volver al menú principal.

U: Volver al menú principal.

Figura 37. Escenario de uso del módulo Propiedades (administrador)

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

U: Información de la propiedad

S: El propietario del local es Felipe. No tiene inquilinos y su coeficiente de reparto es 5.5 para unas dimensiones de 110 metros cuadrados

S: Que desea hacer ahora: repetir información o volver al menú principal

U: Repetir información

S: El propietario del local es Felipe. No tiene inquilinos y su coeficiente de reparto es 5.5 para unas dimensiones de 110 metros cuadrados

S: Que desea hacer ahora: repetir información o volver al menú principal

U: Volver al menú principal

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 38. Escenario de uso del módulo Propiedades (administrado)

4.2.4 Cargos

El módulo de cargos implementa la tercera de las opciones que el sistema ofrece a los usuarios y consiste en una consulta de la composición de la junta de gobierno de la comunidad, es decir, se puede conocer qué propiedad es la que ostenta el cargo que se desee conocer.

En este módulo de consulta de cargos, la información obtenida es la misma para cada tipo de usuario, por lo tanto, el funcionamiento de este módulo será igual a excepción de la arquitectura de cada uno, puesto que las clases que se ejecutan son diferentes para cada tipo de usuario.

ESTRUCTURA Y FUNCIONAMIENTO

Como se ha comentado anteriormente, en este módulo el comportamiento del sistema es similar para cada tipo de usuario. Sin embargo, se van a detallar las arquitecturas de manera independiente para separar las diferentes clases que entran en acción.

- La ejecución del usuario administrador partirá de las opciones ofrecidas por el menú de la clase *MenuAdmin.php*. Al seleccionar la tercera opción, "cargos", se llamará a la clase *Cargos.php*, esta clase preguntará al administrador el cargo que quiere consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada *"gramatica_cargos.xml"* capaz de comprender este cargo. Un ejemplo de esta gramática se puede observar en la Figura 39.

```
<?xml version= "1.0" encoding="UTF-8"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="es-ES" version="1.0" root="MYRULE">
<rule id="MYRULE">
<one-of>
<item>Presidente<tag> out.Cargo="El Presidente es Felipe, propietario del local";</tag> </item>
<item>Vocal<tag> out.Cargo="El vocal es Alberto, propietario del Primero A";</tag> </item>
</one-of>
</rule>
</grammar>
```

Figura 39. Ejemplo de gramatica_cargos.xml

Cuando se haya obtenido el cargo a consultar, la gramática generada, cuyo ejemplo se muestra en la imagen anterior, tendrá asociada una descripción para este cargo, y la clase *Cargos.php* la mostrará para que el interprete de voz la lea.

Tras leer la información del cargo, se ejecutará la clase *CargosOpciones.php*, y ésta, preguntará al administrador que qué desea hacer, ofreciendo un menú con tres opciones: repetir información, consultar otro cargo o volver al menú principal.

En caso de repetir información será la clase *CargosOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las tres opciones a realizar.

Si se decide consultar otro cargo, se volverá a ejecutar la clase *Cargos.php,* que volverá a preguntar por el cargo a consultar y repetirá de nuevo todo el proceso.

Finalmente, en el caso de volver al menú principal, se volverá a llamar a la clase *MenuAdmin.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Cargos para el usuario administrador se muestra en Figura 40.

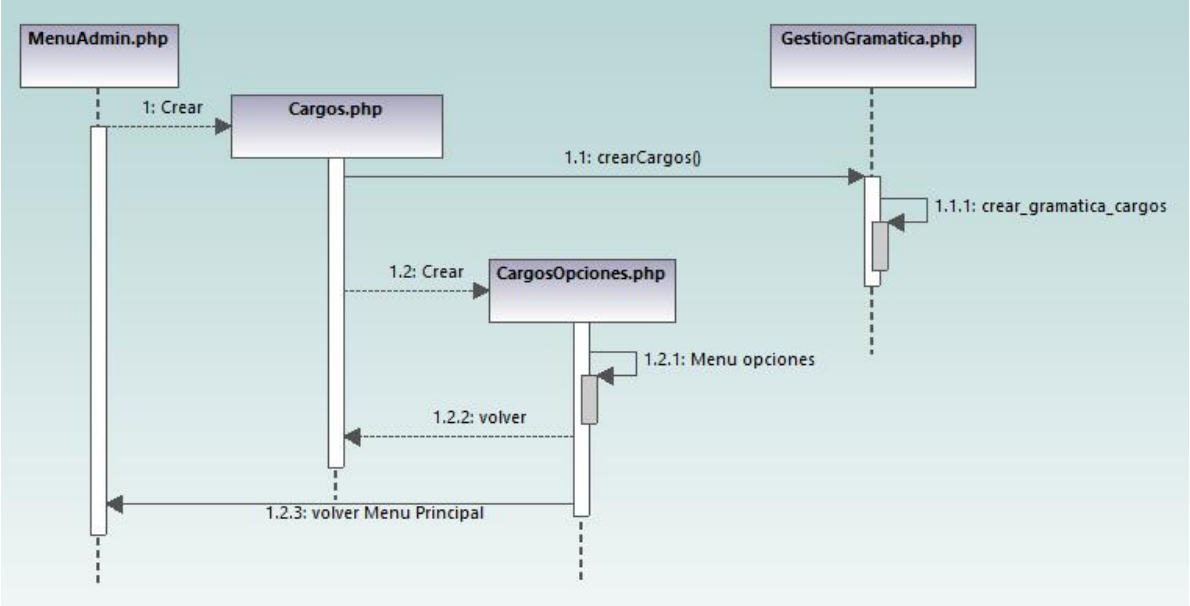

Figura 40. Diagrama de secuencia del módulo Cargos (administrador)

- La ejecución del usuario administrado partirá de las opciones ofrecidas por el menú de la clase *MenuUser.php*. Al seleccionar la tercera opción, "cargos", se llamará a la clase *CargosUsuario.php* que preguntará al usuario por el cargo que quiere consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada *"gramatica_cargos.xml"* capaz de comprender este cargo.

Cuando se haya obtenido el cargo a consultar, la gramática generada, que es la misma que se genera para el administrador, tendrá asociada una descripción para este cargo, y la clase *CargosUsuario.php* la mostrará para que el interprete de voz la lea.

Tras leer la información del cargo, se ejecutará la clase *CargosOpciones.php*, y ésta, preguntará al administrador que qué desea hacer, ofreciendo un menú con 3 opciones: repetir información, consultar otro cargo o volver al menú principal.

En caso de repetir información será la clase *CargosUsuarioOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las tres opciones a realizar.

Si se decide consultar otro cargo, se volverá a ejecutar la clase *CargosUsuario.php,* que volverá a preguntar por el cargo a consultar y repetirá de nuevo todo el proceso.

Finalmente, en el caso de volver al menú principal, se volverá a llamar a la clase *MenuUser.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Cargos para el usuario administrado se muestra en Figura 41.

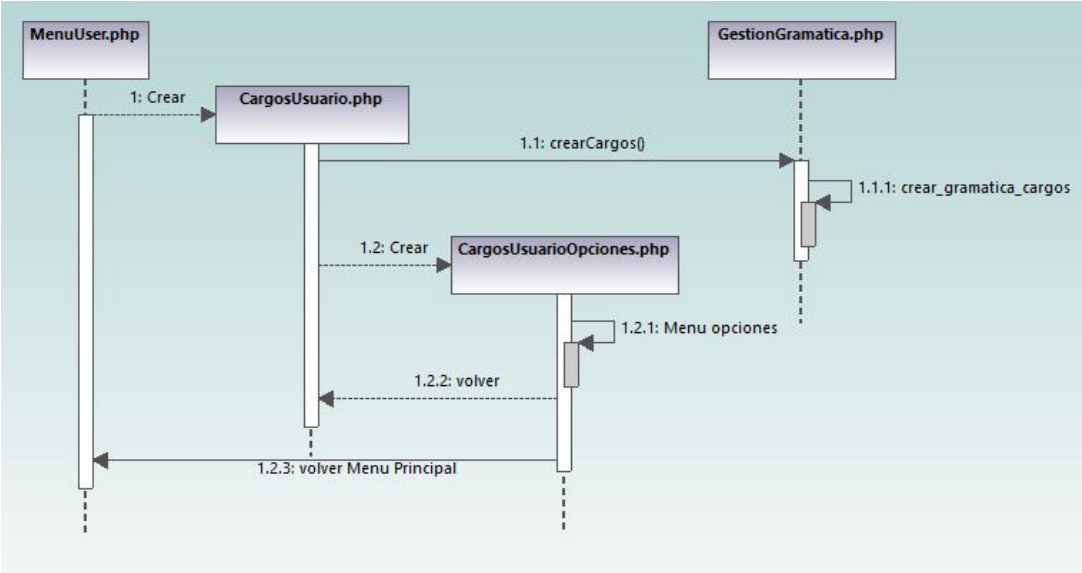

Figura 41. Diagrama de secuencia del módulo Cargos (administrado)

ESCENARIOS DE USO

En las Figuras 42 y 43 se muestra un escenario para cada tipo de usuario.

S:	Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.
U:	Cargos.
S:	Que cargo desea consultar?
U:	Presidente.
S:	El presidente es Felipe, propietario del local.
S:	Que desea hacer ahora: repetir información, consultar otro cargo o volver al menú principal.
U:	Repetir información.
S:	El presidente es Felipe, propietario del local.
S:	Que desea hacer ahora: repetir información, consultar otro piso o volver al menú principal.
U:	Volver al menú principal.
S:	Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 42. Escenario de uso del módulo Cargos (administrador)

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

U: Cargos.

S: Que cargo desea consultar?

U: Presidente.

S: El presidente es Felipe, propietario del local.

S: Que desea hacer ahora: repetir información, consultar otro cargo o volver al menú principal.

U: Consultar otro cargo.

S: Que cargo desea consultar?

U: Vocal.

S: El vocal es Alberto, propietario del primero A

S: Que desea hacer ahora: repetir información, consultar otro piso o volver al menú principal.

U: Volver al menú principal.

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 43. Escenario de uso del módulo Cargos (administrado)

4.2.5 Proveedores

El módulo de proveedores implementa la cuarta de las opciones que el sistema ofrece a los usuarios y permite consultar la información de los proveedores. Esta información se puede obtener consultando un proveedor en concreto y también se puede conocer un listado de los proveedores que tiene registrados la comunidad para realizar un determinado servicio.

En este módulo de consulta de proveedores, al igual que en el módulo anterior de cargos, la información obtenida es la misma para cada tipo de usuario. Es decir, el funcionamiento de este módulo será igual para cada usuario a excepción de la arquitectura de cada uno, puesto que las clases que se ejecutan son diferentes para cada tipo de usuario.

ESTRUCTURA Y FUNCIONAMIENTO

Como se ha comentado anteriormente, en este módulo el comportamiento del sistema es similar para cada tipo de usuario. Sin embargo, se van a detallar las arquitecturas de manera independiente para separar las diferentes clases que entran en acción.

- La ejecución del usuario administrador partirá de las opciones ofrecidas por el menú de la clase *MenuAdmin.php*. Al seleccionar la cuarta opción, "proveedores", se llamará a la clase *Proveedores.php*, esta clase preguntará al administrador si quiere consultar los proveedores por gremio (aquellos que realizan un determinado servicio) o prefiere consultar la información de un proveedor en concreto.

En caso de querer consultar proveedores por gremio, se ejecutará la clase *Gremio.php*. Esta clase solicitará el gremio de los proveedores que se quieren consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada *"gramatica_gremios.xml"* capaz de comprender el gremio solicitado. Un ejemplo de esta gramática se muestra en la Figura 44.

```
<?xml version= "1.0" encoding="UTF-8"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="es-ES" version="1.0" root="MYRULE">
<rule id="MYRULE">
<one-of>
<item>Limpieza<tag> out.Gremio="los proveedores de Limpieza son Anayet, ";</tag> </item>
<item>Jardineria<tag> out.Gremio="los proveedores de Jardineria son Anayet, Roper, ";</tag> </item>
<item>Ascensores<tag> out.Gremio="los proveedores de Ascensores son Endesa, ";</tag> </item>
<item>Seguridad<tag> out.Gremio="los proveedores de Seguridad son AguaLimpia S L, ";</tag> </item>
<item>Piscina<tag> out.Gremio="los proveedores de Mantenimiento piscina son Roper, ";</tag> </item>
</one-of>
</rule>
</grammar>
```

Figura 44. Ejemplo de gramatica_gremios.xml

Cuando se haya obtenido el gremio a consultar, la gramática generada, cuyo ejemplo se muestra en la imagen anterior, tendrá asociada una descripción en la que se indicarán los proveedores dedicados a la actividad indicada. La clase *Gremio.php* mostrará la descripción para que el interprete de voz la lea.

Tras leer el listado de proveedores, se ejecutará la clase *GremiosOpciones.php*, y ésta, preguntará al administrador que qué desea hacer, ofreciendo un menú con 4 opciones: repetir proveedores, consultar proveedores de otro gremio, consultar un proveedor en concreto o volver al menú principal.

En caso de repetir información será la clase *GremiosOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las 4 opciones a realizar.

Si se decide consultar proveedores de otro gremio, se volverá a ejecutar la clase *Gremio.php,* que volverá a preguntar por el gremio a consultar y repetirá de nuevo todo el proceso.

En el caso de que se quiera consultar la información de un proveedor en concreto, la clase que se ejecutará será *InfoProveedor.php.* Esta clase es la misma que se ejecutaría al comienzo del módulo, si se seleccionara la opción de de consultar la información de un proveedor en concreto, cuando la clase *Proveedores.php* ofrece esta opción.

La clase *InfoProveedor.php* preguntará por el proveedor a consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada *"gramatica_proveedores.xml*", capaz de comprender el proveedor solicitado. Un ejemplo de esta gramática se encuentra en la Figura 45.

```
<?xml version= "1.0" encoding="UTF-8"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="es-ES" version="1.0" root="MYRULE">
<rule id="MYRULE">
<one-of>
<item>Anayet<tag> out.Proveedor="El nif del proveedor Anayet es B49008711, y su telefono es el 916005422";</tag> </item>
<item>Endesa<tag> out.Proveedor="El nif del proveedor Endesa es C45221148, y su telefono es el 914658877";</tag> </item>
<item>AguaLimpia S L<tag> out.Proveedor="El nif del proveedor AguaLimpia S L es D49874418, y su telefono es el 976215543";
</tag> </item>
<item>Roper<tag> out.Proveedor="El nif del proveedor Roper es T48875629, y su telefono es el 976014578";</tag> </item>
</one-of>
</rule>
</grammar>
```

Figura 45. Ejemplo de gramatica_proveedores.xml

Cuando se haya obtenido el proveedor a consultar, la gramática generada, cuyo ejemplo se muestra en la imagen anterior, tendrá asociada una descripción en la que se indicará la información a mostrar del proveedor. La clase *InfoProveedor.php* mostrará la descripción para que el interprete de voz la lea.

Tras leer la información del proveedor, se ejecutará la clase *ProveedorOpciones.php*, y ésta, preguntará al administrador que qué desea hacer, ofreciendo un menú con cuatro opciones: repetir información del proveedor, consultar otro proveedor, consultar proveedores de otro gremio o volver al menú principal.

En caso de repetir información será la clase *ProveedorOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las cuatro opciones a realizar.

Si se quiere consultar otro proveedor, se ejecutará de nuevo la clase *InfoProveedor.php*, que volverá a solicitar un proveedor para consultar y repetirá de nuevo el proceso.

Si se decide consultar proveedores de otro gremio, se volverá a ejecutar la clase *Gremio.php,* que volverá a preguntar por el gremio a consultar y repetirá de nuevo todo el proceso explicado con anterioridad.

Finalmente, en el caso de volver al menú principal, se volverá a llamar a la clase *MenuAdmin.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Proveedores para el usuario administrador se muestra en Figura 46.

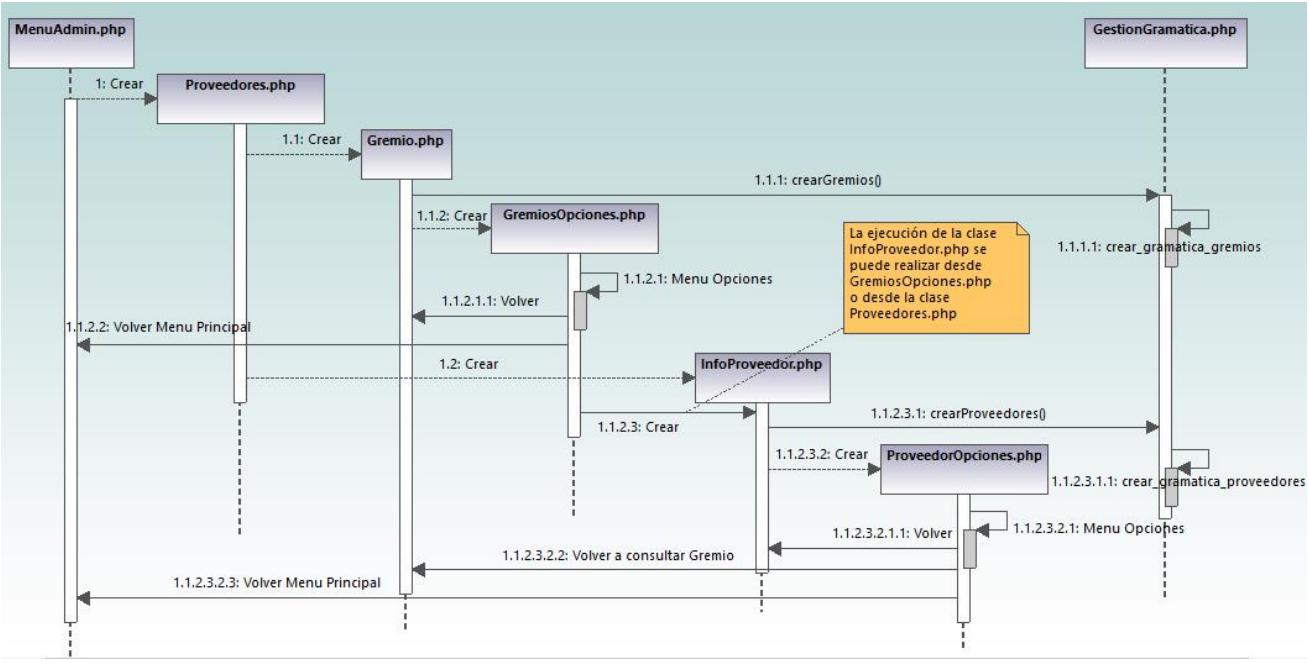

Figura 46. Diagrama de secuencia del módulo Proveedores (administrador)

- La ejecución del usuario administrado tiene un funcionamiento prácticamente igual a la ejecución del administrador, de hecho, la información que se puede consultar es exactamente la misma a la del administrador. Sin embargo, las clases que se ejecutan son diferentes, por lo tanto, a continuación, se volverá a explicar el proceso detalladamente.

Comenzando por las opciones ofrecidas por la clase *MenuUser.php*, al seleccionar la cuarta opción, "proveedores", se llamará a la clase *ProveedoresUsuario.php.* Esta clase preguntará al usuario si quiere consultar los proveedores por gremio o prefiere consultar la información de un proveedor en concreto.

En caso de querer consultar proveedores por gremio, se ejecutará la clase *GremioUsuario.php*. Esta clase solicitará el gremio de los proveedores que se quieren consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada *"gramatica_gremios.xml"* capaz de comprender el gremio solicitado.

Cuando se haya obtenido el gremio a consultar, la gramática generada tendrá asociada una descripción en la que se indicarán los proveedores dedicados a dicha actividad. La clase *GremioUsuario.php* mostrará la descripción para que el interprete de voz la lea.

Tras leer el listado de proveedores, se ejecutará la clase *GremiosUsuarioOpciones.php*, y ésta, preguntará al usuario que qué desea hacer, ofreciendo un menú con cuatro opciones: repetir proveedores, consultar proveedores de otro gremio, consultar un proveedor en concreto o volver al menú principal.

En caso de repetir información será la clase *GremiosUsuariosOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las cuatro opciones a realizar.

Si se decide consultar proveedores de otro gremio, se volverá a ejecutar la clase *GremioUsuario.php,* que volverá a preguntar por el gremio a consultar y repetirá de nuevo todo el proceso.

En el caso de que se quiera consultar la información de un proveedor en concreto, la clase que se ejecutará será *InfoProveedorUsuario.php.* Esta clase es la misma que se ejecutaría al comienzo del módulo, si se seleccionara la opción de de consultar la información de un proveedor en concreto, cuando la clase *ProveedoresUsuario.php* ofrece esta opción.

La clase *InfoProveedorUsuario.php* preguntará por el proveedor a consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada *"gramatica_proveedores.xml",* capaz de comprender el proveedor solicitado.

Cuando se haya obtenido el proveedor a consultar, la gramática generada tendrá asociada una descripción en la que se indicará la información a mostrar del proveedor. La clase *InfoProveedorUsuario.php* mostrará la descripción para que el interprete de voz la lea.

Tras leer la información del proveedor, se ejecutará la clase *ProveedorUsuarioOpciones.php,* y ésta, preguntará al usuario administrado que qué desea hacer, ofreciendo un menú con cuatro opciones: repetir información del proveedor, consultar otro proveedor, consultar proveedores de otro gremio o volver al menú principal.

En caso de repetir información será la clase *ProveedorUsuarioOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las cuatro opciones a realizar.

Si se quiere consultar otro proveedor, se ejecutará de nuevo la clase *InfoProveedorUsuario.php,* que volverá a solicitar un proveedor para consultar y repetirá de nuevo el proceso.

Si se decide consultar proveedores de otro gremio, se volverá a ejecutar la clase *GremioUsuario.php,* que volverá a preguntar por el gremio a consultar y repetirá de nuevo todo el proceso explicado con anterioridad.

Finalmente, en el caso de volver al menú principal, se volverá a llamar a la clase *MenuUser.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Proveedores para el usuario administrado se muestra en Figura 47.

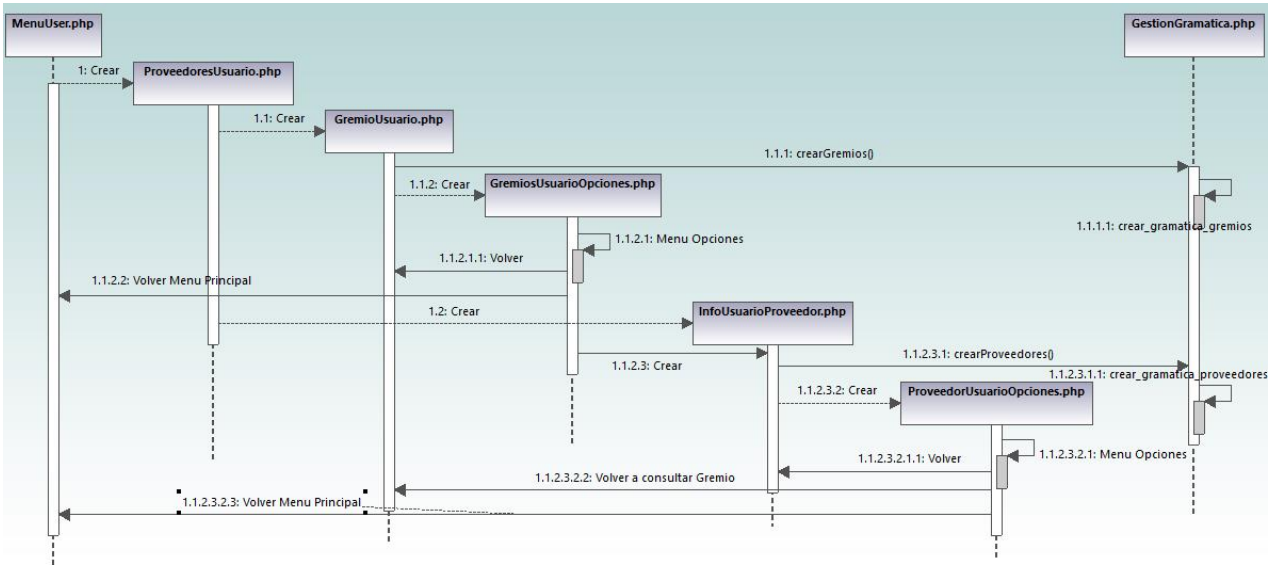

Figura 47. Diagrama de secuencia del módulo Proveedores (administrado)

ESCENARIOS DE USO

En las Figuras 48 y 49 se muestra un escenario para cada tipo de usuario:

S: Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

U: Proveedores.

S: Que desea hacer: consultar proveedores por gremio o consultar información de un proveedor en concreto.

U: Consultar información de un proveedor en concreto.

S: Que proveedor desea consultar.

U: Endesa.

S: El NIF del proveedor Endesa es C45221148, y su teléfono es el 914658877

S: Que desea hacer ahora: repetir información, consultar otro proveedor, consultar proveedores de un gremio o volver al menú principal.

S: Volver al menú principal

S: Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 48. Escenario de uso del módulo Proveedores (administrador)

S:	Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.
U:	Proveedores.
S:	Que desea hacer: consultar proveedores por gremio o consultar información de un proveedor en concreto.
U:	Consultar proveedores por gremio.
S:	Los proveedores de qué gremio desea obtener.
U:	Jardinería.
S:	Los proveedores de Jardinería son Anayet, Roper
S:	Que desea hacer ahora: repetir proveedores, consultar proveedores de otro gremio, consultar un proveedor en concreto o volver al menú principal.
U:	Volver al menú principal
S:	Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 49. Escenario de uso del módulo Proveedores (administrado)

4.2.6 Contratos

El módulo de contratos implementa la quinta de las opciones que el sistema ofrece a los usuarios y consiste en una consulta de la información de los contratos que la comunidad tiene vigentes.

En este módulo de consulta de contratos, la información obtenida es la misma para cada tipo de usuario, por lo tanto, el funcionamiento de este módulo será igual a excepción de la arquitectura de cada uno, puesto que las clases que se ejecutan son diferentes para cada tipo de usuario.

ESTRUCTURA Y FUNCIONAMIENTO

Como se ha comentado anteriormente, en este módulo el comportamiento del sistema es similar para cada tipo de usuario. Sin embargo, se van a detallar las arquitecturas de manera independiente para separar las diferentes clases que entran en acción.

- La ejecución del usuario administrador partirá de las opciones ofrecidas por el menú de la clase *MenuAdmin.php*. Al seleccionar la quinta opción, “contratos”, se

llamará a la clase *Contratos.php*. Esta clase solicitará al administrador el contrato que quiere consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada *"gramatica_contratos.xml"* capaz de comprender este contrato. Un ejemplo de esta gramática se puede ver en la Figura 50.

```
<?xml version= "1.0" encoding="UTF-8"?>
<grammar xmlns="http://www.w3.org/2001/06/grammar" xml:lang="es-ES" version="1.0" root="MYRULE">
<rule id="MYRULE">
<one-of>
<item>Ascensores<tag> out.Contrato="El contrato de Ascensores lo tiene Roper por 15000 euros. Comienza el 01 de enero de
2017 y termina el 01 de enero de 2018";</tag> </item>
<item>Limpieza<tag> out.Contrato="El contrato de Limpieza lo tiene Anayet por 12000 euros. Comienza el 21 de septiembre de
2016 y termina el 21 de septiembre de 2018";</tag> </item>
<item>Piscina<tag> out.Contrato="El contrato de Piscina lo tiene Anayet por 5000 euros. Comienza el 01 de junio de 2017 y
termina el 30 de septiembre de 2017";</tag> </item>
</one-of>
</rule>
</grammar>
```

Figura 50. Ejemplo de gramatica_contratos.xml

Cuando se haya obtenido el contrato a consultar, la gramática generada, cuyo ejemplo se muestra en la imagen anterior, tendrá asociada una descripción para este contrato, y la clase *Contratos.php* la mostrará para que el interprete de voz la lea.

Tras leer la información del contrato, se ejecutará la clase *ContratoOpciones.php*, y ésta, preguntará al administrador que qué desea hacer, ofreciendo un menú con tres opciones: repetir información, consultar otro contrato o volver al menú principal.

En caso de repetir información será la clase *ContratoOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las tres opciones a realizar.

Si se decide consultar otro contrato, se volverá a ejecutar la clase *Contratos.php*, que volverá a preguntar por el contrato a consultar y repetirá de nuevo todo el proceso.

Finalmente, en el caso de volver al menú principal, se volverá a llamar a la clase *MenuAdmin.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Contratos para el usuario administrador se muestra en Figura 51.

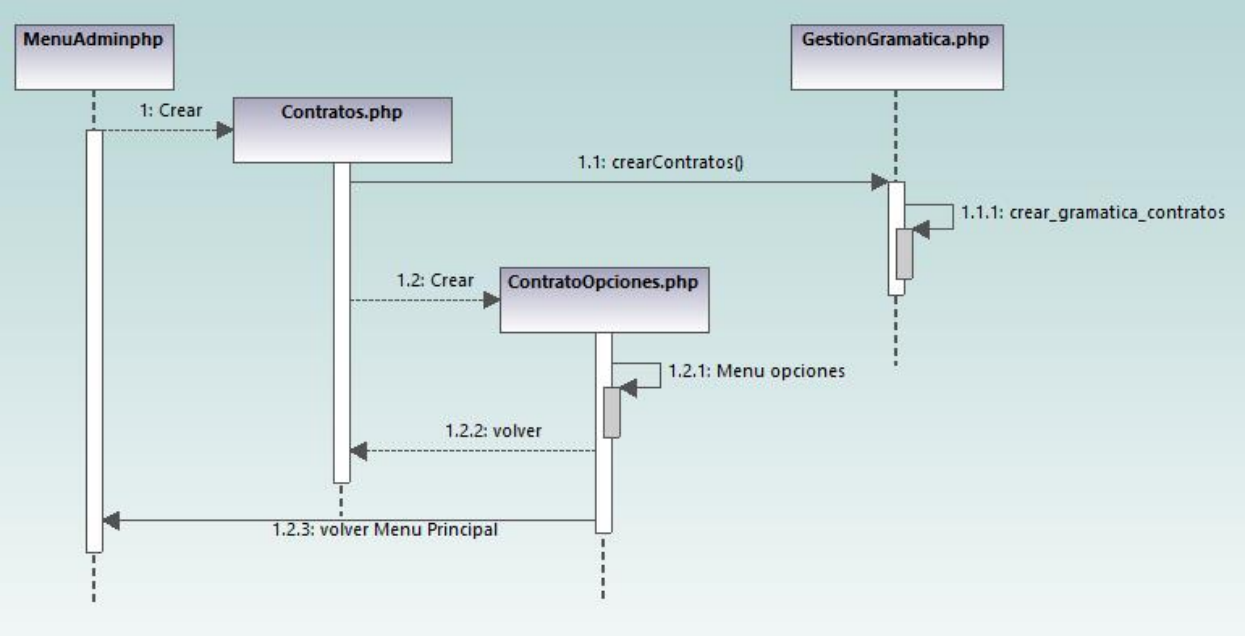

Figura 51. Diagrama de secuencia del módulo Contratos (administrador)

- La ejecución del usuario administrado partirá de las opciones ofrecidas por el menú de la clase *MenuUser.php*. Al seleccionar la tercera opción, "contratos", se llamará a la clase *ContratosUsuario.php* que preguntará al usuario por el contrato que quiere consultar y llamará a la clase *GestionGramatica.php* para que genere una gramática, llamada *"gramatica_contratos.xml"* capaz de comprender este cargo.

Cuando se haya obtenido el contrato a consultar, la gramática generada, que es la misma que se genera para el administrador, tendrá asociada una descripción para este contrato, y la clase *ContratosUsuario.php* la mostrará para que el interprete de voz la lea.

Tras leer la información del contrato, se ejecutará la clase *ContratoUsuarioOpciones.php*, y ésta, preguntará al administrador que qué desea hacer, ofreciendo un menú con tres opciones: repetir información, consultar otro contrato o volver al menú principal.

En caso de repetir información será la clase *ContratoUsuarioOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las tres opciones a realizar.

Si se decide consultar otro cargo, se volverá a ejecutar la clase *ContratosUsuario.php,* que volverá a preguntar por el cargo a consultar y repetirá de nuevo todo el proceso.

Finalmente, en el caso de volver al menú principal, se volverá a llamar a la clase *MenuUser.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Contratos para el usuario administrado se muestra en Figura 52.

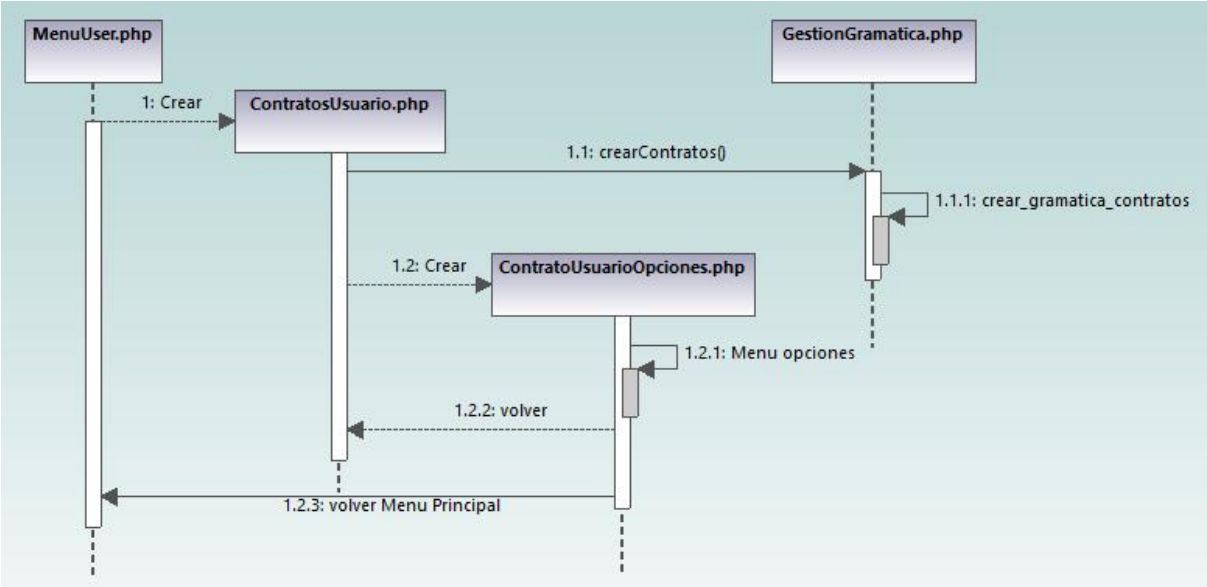

Figura 52. Diagrama de secuencia del módulo Contratos (administrado)

ESCENARIOS DE USO

En las Figuras 53 y 54 se muestra un escenario para cada tipo de usuario:

S:	Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.
U:	Contratos.
S:	Que contrato desea consultar.
U:	Limpieza.
S:	El contrato de Limpieza lo tiene Anayet por 12000 euros. Comienza el 21 de septiembre de 2016 y termina el 21 de septiembre de 2018
S:	Que desea hacer ahora: repetir información, consultar otro contrato o volver al menú principal.
U:	Repetir información
S:	El contrato de Limpieza lo tiene Anayet por 12000 euros. Comienza el 21 de septiembre de 2016 y termina el 21 de septiembre de 2018
S:	Que desea hacer ahora: repetir información, consultar otro contrato o volver al menú principal.
U:	Volver al menú principal.
S:	Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 53. Escenario de uso del módulo Contratos (administrador)

S:	Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.
U:	Contratos.
S:	Que contrato desea consultar.
U:	Piscina.
S:	El contrato de Piscina lo tiene Anayet por 5000 euros. Comienza el 01 de junio de 2017 y termina el 30 de septiembre de 2017.
S:	Que desea hacer ahora: repetir información, consultar otro contrato o volver al menú principal.
U:	Consultar otro contrato
S:	Que contrato desea consultar.
U:	Ascensores.
S:	El contrato de Ascensores lo tiene Roper por 15000 euros. Comienza el 01 de enero de 2017 y termina el 01 de enero de 2018.
S:	Que desea hacer ahora: repetir información, consultar otro contrato o volver al menú principal.
U:	Volver al menú principal.
S:	Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 54. Escenario de uso del módulo Contratos (administrado)

4.2.7 Juntas Vecinales

El módulo de juntas vecinales implementa la sexta de las opciones que el sistema ofrece a los usuarios y permite consultar información de las juntas de vecinos celebradas en una comunidad.

En este módulo de consulta de juntas vecinales, los dos tipos de usuarios podrán consultar las actas de las juntas celebradas y también el orden del día, tanto de las juntas celebradas como de las que están planificadas pero pendientes de celebración. Además de estas 2 opciones, el administrador tendrá la opción de realizar un envío masivo de correos electrónicos, a los usuarios de una comunidad, informando de la fecha de celebración de una junta vecinal futura.

ESTRUCTURA Y FUNCIONAMIENTO

En este módulo existen diferentes comportamientos dependiendo del tipo de usuario, por lo tanto, se van a detallar las arquitecturas de cada usuario de forma independiente para separar las diferentes clases que entran en acción.

- La ejecución del administrador partirá de las opciones ofrecidas por el menú de la clase *MenuAdmin.php*. Al seleccionar la sexta opción, "juntas vecinales", se llamará a la clase *Juntas.php,* esta clase preguntará al administrador si quiere consultar el orden del día de una junta, el acta de una junta, enviar un correo a los propietarios o si prefiere no consultar nada y volver a menú principal.

En caso de querer consultar el orden del día de una junta, se ejecutará la clase *OrdenJunta.php*. Esta clase solicitará al administrador que marque en su teléfono la fecha de la junta que desea consultar. En este caso no se creará ningún fichero con la gramática que valide la petición del usuario, en su lugar, se indicará que la petición esperada será de tipo fecha y DMTF, es decir, que se introducirá mediante las pulsaciones de las teclas del teléfono. En la Figura 55 se muestra el código que define esta gramática.

```
<field name="Orden">

    <prompt bargein="false">Marque en su telefono la fecha de la junta que desea consultar</prompt>
    <grammar src="builtin:dtmf/date"/>

   <!-- The user was silent, restart the field. -->
    <noinput>
        No he entendido la fecha. Por favor, intentelo de nuevo.
        <reprompt/>
    </noinput>

  <!-- The user said something that was not defined in our grammar. -->
    <nomatch>
        No existe ninguna junta en esa fecha. Intentelo de nuevo.
        <reprompt/>
    </nomatch>

</field>
```

Figura 55. Código que recoge la fecha indicada por el usuario

Cuando se haya obtenido una fecha con formato válida, se llamará a la clase *CompruebaFechaOrden.php,* que a su vez llamará a la clase *GestionGramatica.php* con el objetivo de comprobar si la fecha obtenida coincide con alguna de las fechas de juntas de vecinos que hay registradas.

1. Si la fecha existe, se volverá a llamar a la clase *GestionGramatica.php,* esta vez para obtener el orden del día de la junta de esa fecha. Esa orden del día será mostrada por la clase *CompruebaFechaOrden.php* y porteriormente se volverá a ejecutar a la clase *Juntas.php,* que ofrecerá las opciones iniciales del módulo y se repetirá el proceso.

2. Si la fecha no existe, la clase *CompruebaFechaOrden.php* indicará que esa fecha no existe y volverá a ejecutar la clase *OrdenJunta.php* para que se vuelva a solicitar una nueva fecha y repetir el proceso.

Si tras escuchar las opciones iniciales, se desea consultar el acta de una junta, el proceso será similar al de consulta de las órdenes del día. Inicialmente se ejecutará la

clase *ActaJunta.php* que solicitará al administrador que marque en su teléfono la fecha de la junta que desea consultar. En este caso también se indicará que la petición esperada será de tipo fecha y DMTF.

Cuando se haya obtenido una fecha con formato válida, se llamará a la clase *CompruebaFechaActa.php,* que a su vez llamará a la clase *GestionGramatica.php* con el objetivo de comprobar si la fecha obtenida coincide con alguna de las fechas de juntas de vecinos que hay registradas.

1. Si la fecha existe, se volverá a llamar a la clase *GestionGramatica.php,* esta vez para obtener el acta de la junta de esa fecha. El acta será mostrada por la clase *CompruebaFechaActa.php* y porteriormente se volverá a ejecutar a la clase *Juntas.php,* que ofrecerá las opciones iniciales del módulo y se repetirá el proceso.

2. Si la fecha no existe, la clase *CompruebaFechaActa.php* indicará que esa fecha no existe y volverá a ejecutar la clase *ActaJunta.php* para que se vuelva a solicitar una nueva fecha y repetir el proceso.

En el caso de que, en las opciones iniciales, se desee enviar un correo a los propietarios de la comunidad, se ejecutará la clase *CorreoJuntas.php.* Esta clase llamará a la clase GestionGramatica.php y consultará todos los correos de los propietarios de la comunidad, enviándoles un correo electrónico informando de la fecha de la próxima junta de vecinos. Un ejemplo de este correo electrónico se puede observar en la Figura 56.

Figura 56. Correo electrónico con la fecha de próxima junta

Finalmente, si lo que se desea en las opciones iniciales es volver al menú principal, se volverá a llamar a la clase *MenuAdmin.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Juntas Vecinales para el usuario administrador se muestra en Figura 57.

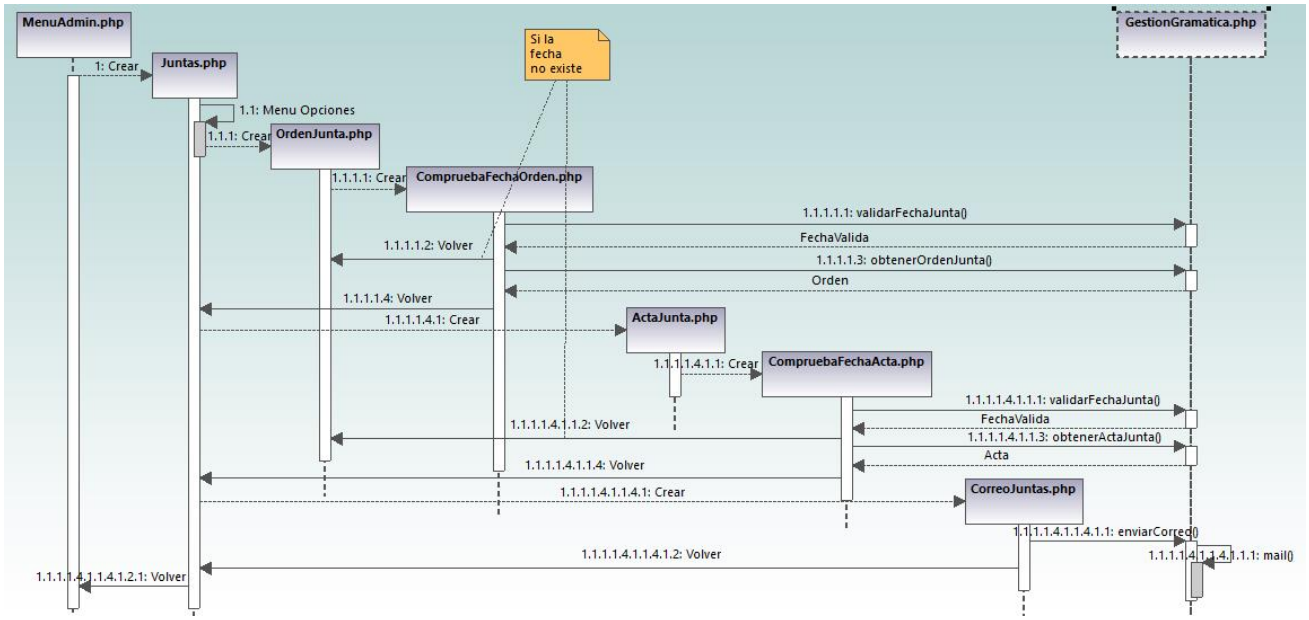

Figura 57. Diagrama de secuencia del módulo Juntas Vecinales (administrador)

- La ejecución del usuario administrado partirá de las opciones ofrecidas por el menú de la clase *MenuUser.php*. Al seleccionar la sexta opción, "juntas vecinales", se llamará a la clase *JuntasUsuario.php,* esta clase preguntará al usuario si quiere consultar el orden del día de una junta, consultar el acta de una junta o volver a menú principal.

En caso de querer consultar el orden del día de una junta, se ejecutará la clase *OrdenJuntaUsuario.php*. Esta clase solicitará al usuario que marque en su teléfono la fecha de la junta que desea consultar. En esta gramática se indicará que la petición esperada será de tipo fecha y DMTF.

Cuando se haya obtenido una fecha con formato válida, se llamará a la clase *CompruebaFechaOrdenUsuario.php*, que a su vez llamará a la clase *GestionGramatica.php* con el objetivo de comprobar si la fecha obtenida coincide con alguna de las fechas de juntas de vecinos que hay registradas.

1. Si la fecha existe, se volverá a llamar a la clase *GestionGramatica.php,* esta vez para obtener el orden del día de la junta de esa fecha. Esa orden del día será mostrada por la clase *CompruebaFechaOrdenUsuario.php* y porteriormente se volverá a ejecutar a la clase *JuntasUsuario.php,* que ofrecerá las opciones iniciales del módulo y se repetirá el proceso.

2. Si la fecha no existe, la clase *CompruebaFechaOrdenUsuario.php* indicará que esa fecha no existe y volverá a ejecutar la clase *OrdenJuntaUsuario.php* para que se vuelva a solicitar una nueva fecha y repetir el proceso.

Si tras escuchar las opciones iniciales, se desea consultar el acta de una junta, el proceso será similar al de consulta de las órdenes del día. Inicialmente se ejecutará la clase *ActaJuntaUsuario.php* que solicitará al usuario que marque en su teléfono la fecha de la junta que desea consultar. En este caso también se indicará que la petición esperada será de tipo fecha y DMTF.

Cuando se haya obtenido una fecha con formato válida, se llamará a la clase *CompruebaFechaActaUsuario.php*, que a su vez llamará a la clase *GestionGramatica.php* con el objetivo de comprobar si la fecha obtenida coincide con alguna de las fechas de juntas de vecinos que hay registradas.

1. Si la fecha existe, se volverá a llamar a la clase *GestionGramatica.php*, esta vez para obtener el acta de la junta de esa fecha. El acta será mostrada por la clase *CompruebaFechaActaUsuario.php* y porteriormente se volverá a ejecutar a la clase *JuntasUsuario.php*, que ofrecerá las opciones iniciales del módulo y se repetirá el proceso.

2. Si la fecha no existe, la clase *CompruebaFechaActaUsuario.php* indicará que esa fecha no existe y volverá a ejecutar la clase *ActaJuntaUsuario.php* para que se vuelva a solicitar una nueva fecha y repetir el proceso.

Finalmente, si lo que se desea en las opciones iniciales es volver al menú principal, se volverá a llamar a la clase *MenuUser.php* que volverá a ofrecer las opciones del menú.

El diagrama de secuencia de este módulo de Juntas Vecinales para el usuario administrado se muestra en Figura 58.

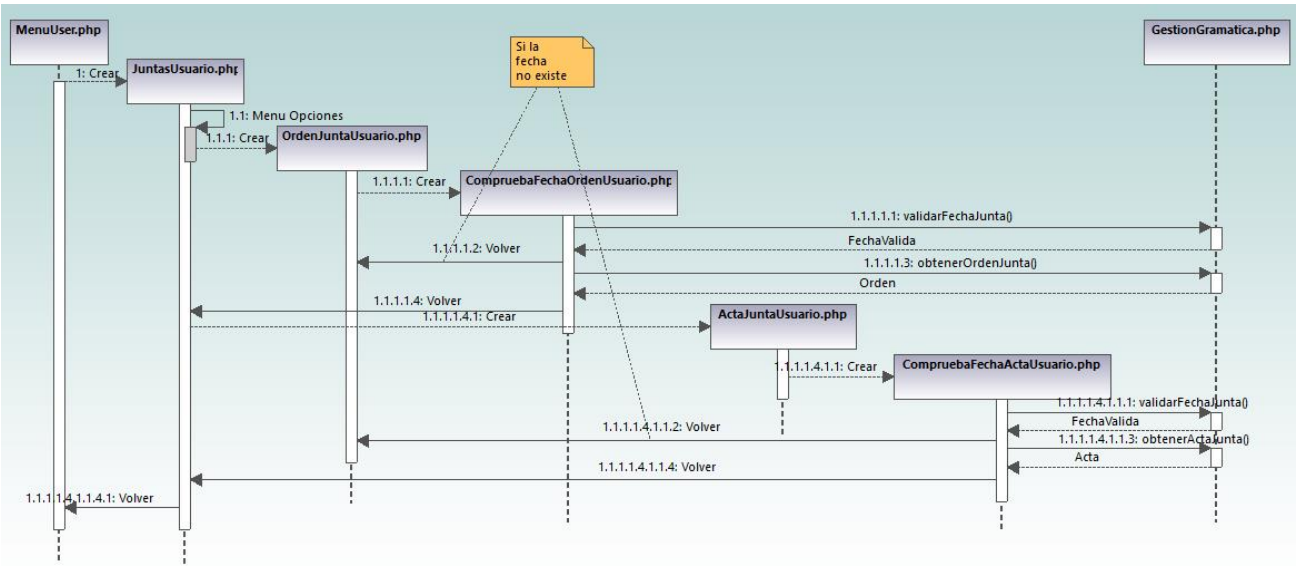

Figura 58. Diagrama de secuencia del módulo Juntas Vecinales (administrado)

ESCENARIOS DE USO

En las Figuras 59 y 60 se muestra un escenario para cada tipo de usuario:

S: Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

U: Juntas vecinales.

S: Que desea hacer: consultar orden del día de una junta, consultar acta de una junta, enviar correo a los propietarios o volver al menú principal.

U: Consultar orden del día de una junta

S: Marque en su teléfono la fecha de la junta que desea consultar

U: 20170328

S: 1. Elección de presidente. 2. Contrato de limpieza. 3. Ruegos y preguntas

S: Que desea hacer: consultar orden del día de una junta, consultar acta de una junta, enviar correo a los propietarios o volver al menú principal.

U: Enviar correo a los propietarios.

S: Correos enviados

S: Que desea hacer: consultar orden del día de una junta, consultar acta de una junta, enviar correo a los propietarios o volver al menú principal.

U: Volver al menú principal.

S: Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 59. Escenario de uso del módulo Juntas Vecinales (administrador)

S:	Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.
U:	Juntas vecinales.
S:	Que desea hacer: consultar orden del día de una junta, consultar acta de una junta o volver al menú principal.
U:	Consultar acta de una junta.
S:	Marque en su teléfono la fecha de la junta que desea consultar
U:	20170328
S:	Manuel se ofrece voluntario para ser presidente durante el primer año. Se respetará el contrato de limpieza con Anayet, pero en la resolución del mismo, se buscará otra empresa porque no se está contento con su trabajo…
S:	Que desea hacer: consultar orden del día de una junta, consultar acta de una junta, enviar correo a los propietarios o volver al menú principal.
U:	Volver al menú principal
S:	Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 60. Escenario de uso del módulo Juntas Vecinales (administrado)

4.2.8 Contabilidad

El módulo de contabilidad implementa la última de las opciones que el sistema ofrece a los usuarios y permite consultar varios aspectos de la contabilidad de la comunidad, tales como el saldo actual de la cuenta, los últimos movimientos realizados o incluso los recibos pendientes de pago.

En este módulo de consulta de contabilidad los dos tipos de usuarios pueden consultar el estado actual y los últimos movimientos de la cuenta y la información ofrecida será la misma para los dos. Sin embargo, en la consulta de recibos pendientes el administrador recibirá información resumida de todos los recibos que hay pendientes, mientras que el administrado solo puede consultar sus recibos pendientes en el caso de que tenga alguno.

ESTRUCTURA Y FUNCIONAMIENTO

En este módulo, el comportamiento del sistema es similar para cada tipo de usuario, sin embargo, se van a detallar las arquitecturas de manera independiente para separar las diferentes clases que entran en acción.

- La ejecución del usuario administrador partirá de las opciones ofrecidas por el menú de la clase *MenuAdmin.php*. Al seleccionar la séptima y última opción, "contabilidad", se llamará a la clase *Contabilidad.php*. Esta clase preguntará al administrador si quiere consultar el saldo actual, consultar los últimos movimientos, los recibos pendientes o volver al menú principal.

En caso de querer consultar el saldo actual, se ejecutará la clase *Saldo.php*. Esta clase mostrará el saldo actual de la cuenta, para lo cual, llamará a la clase *GestionGramatica.php* que realizará la consulta correspondiente y devolverá la información de este saldo a *Saldo.php*.

Tras mostrar el saldo actual, se volverá a ejecutar la clase *Contabilidad.php* para volver a ofrecer las opciones iniciales.

Si lo que se desea consultar son los últimos movimientos de la cuenta, se ejecutará la clase *Movimientos.php*. Esta clase, solicitará al administrador el número de movimientos que desea consultar, y para comprenderlos, generará una gramática estática DMTF en la que permitirá consultar entre 0 y 10 movimientos. El código que define esta gramática en el sistema se puede observar en la Figura 61.

```
<field name="Movimiento">

    <prompt bargein="false">Marque el numero de movimientos que desea consultar</prompt>
    <grammar xml:lang="es-ES" root = "MYRULE" mode="dtmf">
        <rule id="MYRULE" scope = "public">
        <one-of>
            <item> 1 </item>
            <item> 2 </item>
            <item> 3 </item>
            <item> 4 </item>
            <item> 5 </item>
            <item> 6 </item>
            <item> 7 </item>
            <item> 8 </item>
            <item> 9 </item>
        </one-of>
        </rule>
    </grammar>

   <!-- The user was silent, restart the field. -->
    <noinput>
        No he entendido cuantos movimientos. Por favor, intentelo de nuevo.
        <reprompt/>
    </noinput>

  <!-- The user said something that was not defined in our grammar. -->
    <nomatch>
        No es un numero entre 0 y 10. Intentelo de nuevo.
        <reprompt/>
    </nomatch>

</field>
```

Figura 61. Código que genera gramática para el número de movimientos

Una vez se sepa el número de movimientos a consultar, se ejecutará la clase *ObtieneMovimientos.php*, y ésta llamará a *GestionGramática.php* para que realice la consulta y le devuelva la información. Una vez tenga la información deseada, la clase *ObtieneMovimientos.php* la mostrará para que la lea el intérprete de voz.

Tras mostrar esta información, se volverá a ejecutar la clase *Contabilidad.php* para volver a ofrecer las opciones iniciales.

En el caso de preferir consultar los recibos pendientes, será la propia clase *Contabilidad.php*, la que llame a *GestionGramtica.php* solicitando estos recibos pendientes y los mostrará una vez los haya recibido.

Después de mostrar la información de los recibos pendiente, se llamará a la clase *RecibosOpciones.php* para que se ofrezca al administrador un menú con nuevas opciones para realizar: repetir recibos, volver al menú de contabilidad o volver al menú principal.

En caso de repetir información será la clase *RecibosOpciones.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las tres opciones a realizar.

Si se decide volver al menú de contabilidad, se volverá a ejecutar la clase *Contabilidad.php*, que volverá a ofrecer el primer menú y repetirá de nuevo todo el proceso. Sin embargo, si se decide por volver al menú principal, donde se volverá, será a llamar a la clase *MenuAdmin.php* que volverá a ofrecer las opciones del menú principal.

La clase *MenuAdmin.php* también será la que se ejecute si se desea volver al menú principal desde la clase *Contabilidad.php*.

El diagrama de secuencia de este módulo de Contabilidad para el usuario administrador se muestra en Figura 62.

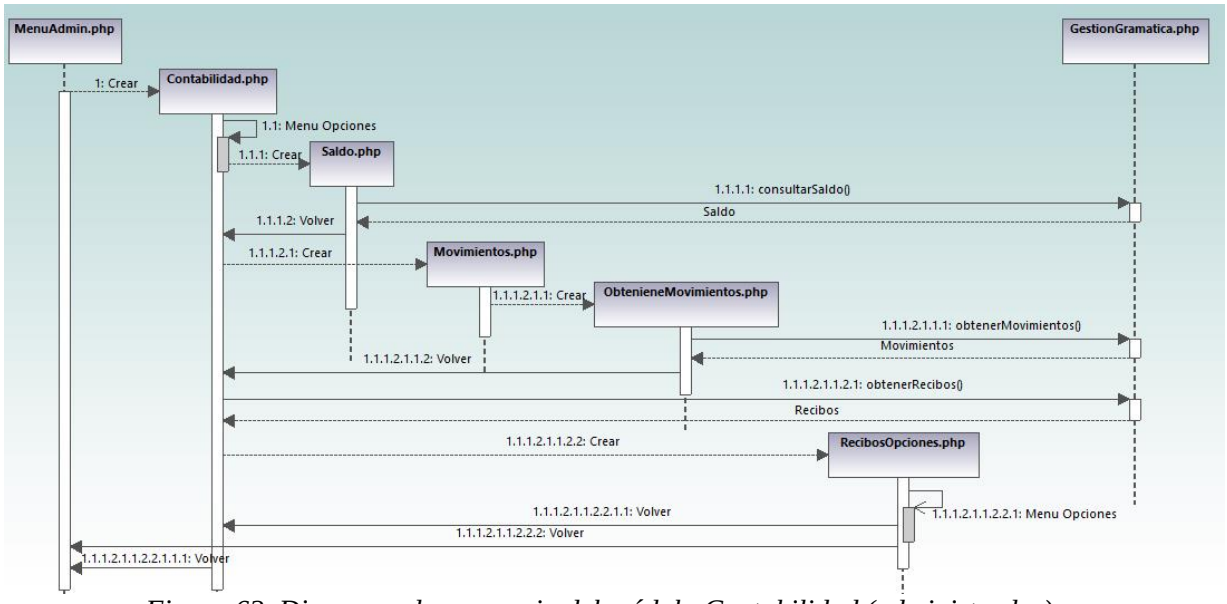

Figura 62. Diagrama de secuencia del módulo Contabilidad (administrador)

- La ejecución del usuario administrado partirá de las opciones ofrecidas por el menú de la clase *MenuUser.php*. Al seleccionar la última opción, "contabilidad", se llamará a la clase *ContabilidadUsuario.php.* Esta clase preguntará al usuario si quiere consultar el saldo actual, consultar los últimos movimientos, los recibos pendientes o volver al menú principal.

En caso de querer consultar el saldo actual, se ejecutará la clase *SaldoUsuario.php*. Esta clase mostrará el saldo actual de la cuenta, para lo cual, llamará a la clase *GestionGramatica.php* que realizará la consulta correspondiente y devolverá la información de este saldo a *SaldoUsuario.php*.

Tras mostrar el saldo actual, se volverá a ejecutar la clase *ContabilidadUsuario.php* para volver a ofrecer las opciones iniciales.

Si lo que se desea consultar son los últimos movimientos de la cuenta, se ejecutará la clase *MovimientosUsuario.php*. Esta clase, solicitará al usuario el número de movimientos que desea consultar, y para comprenderlos, generará una gramática estática DMTF en la que permitirá consultar entre 0 y 10 movimientos.

Una vez se sepa el número de movimientos a consultar, se ejecutará la clase *ObtieneMovimientosUsuario.php,* y ésta llamará a *GestionGramática.php* para que realice la consulta y le devuelva la información. Una vez tenga la información deseada, la clase *ObtieneMovimientosUsuario.php* la mostrará para que la lea el intérprete de voz.

Tras mostrar esta información, se volverá a ejecutar la clase *ContabilidadUsuario.php* para volver a ofrecer las opciones iniciales.

En el caso de preferir consultar los recibos pendientes, será la propia clase *ContabilidadUsuario.php,* la que llame a *GestionGramtica.php* solicitando los recibos pendientes del usuario en caso de tenerlos, y los mostrará una vez los haya recibido.

Después de mostrar la información de los recibos pendiente, se llamará a la clase *RecibosOpcionesUsuario.php* para que se ofrezca al usuario un menú con nuevas opciones para realizar: repetir infirmación, volver al menú de contabilidad o volver al menú principal.

En caso de repetir información será la clase *RecibosOpcionesUsuario.php* la encargada de repetirla tantas veces como se quiera, puesto que después de cada repetición de la información se vuelve a preguntar por las tres opciones a realizar.

Si se decide volver al menú de contabilidad, se volverá a ejecutar la clase *ContabilidadUsuario.php,* que volverá a ofrecer el primer menú y repetirá de nuevo todo el proceso. Sin embargo, si se decide por volver al menú principal, donde se volverá, será a llamar a la clase *MenuUser.php* que volverá a ofrecer las opciones del menú principal.

La clase *MenuUser.php* también será la que se ejecute si se desea volver al menú principal desde la clase *ContabilidadUsuario.php.*

El diagrama de secuencia de este módulo de Contabilidad para el usuario administrado se muestra en Figura 63.

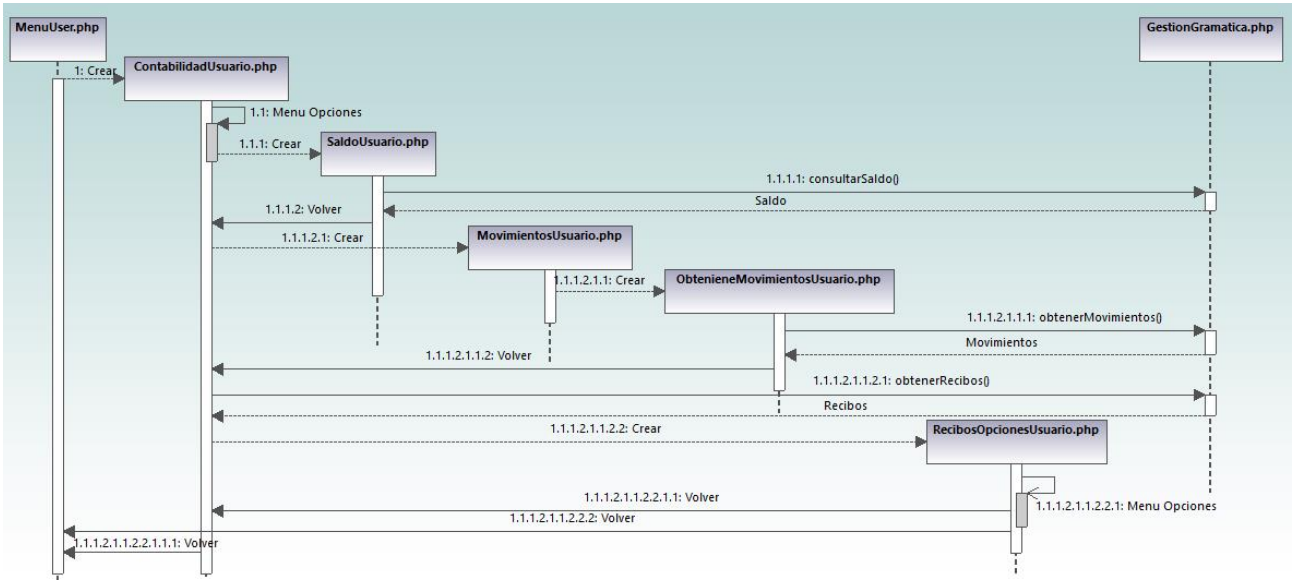

Figura 63. Diagrama de secuencia del módulo Contabilidad (administrado)

ESCENARIOS DE USO

En las Figuras 64 y 65 se muestra un escenario para cada tipo de usuario.

S:	Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.
U:	Contabilidad.
S:	Que desea consultar: saldo actual, últimos movimientos, recibos pendientes o volver al menú principal.
U:	Saldo actual
S:	El saldo actual de la cuenta es 525.25 euros.
S:	Que desea consultar: saldo actual, últimos movimientos, recibos pendientes o volver al menú principal.
U:	Recibos pendientes.
S:	Hay un total de 2 recibos pendientes, con un valor de 350 euros.
S:	Que desea hacer ahora: repetir recibos, volver al menú de contabilidad o volver al menú principal.
U:	Volver al menú principal.
S:	Bienvenido al menú de administración de la comunidad Burgos. Que desea consultar: Descripción, Propiedades, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 64. Escenario de uso del módulo Contabilidad (administrador)

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

U: Contabilidad.

S: Que desea consultar: saldo actual, últimos movimientos, recibos pendientes o volver al menú principal.

U: Últimos movimientos.

S: Marque el número de movimientos que desea consultar.

U: 2.

S: 28 de agosto de 2017. Ingreso cuota Felipe. Importe 150 euros. 30 de agosto de 2017. Pago mantenimiento ascenso. Importe 80 euros.

S: Que desea consultar: saldo actual, últimos movimientos, recibos pendientes o volver al menú principal.

U: Recibos pendientes.

S: No tiene recibos pendientes.

S: Que desea hacer ahora: repetir recibos, volver al menú de contabilidad o volver al menú principal.

U: Volver al menú principal

S: Bienvenido al menú de usuario de la comunidad Burgos. Que desea consultar: información de la comunidad, información de la Propiedad, Cargos, Proveedores, Contratos, Juntas vecinales o Contabilidad.

Figura 65. Escenario de uso del módulo Juntas Vecinales (administrado)

Capítulo 5

Evaluación del sistema

En este capítulo se describe la evaluación que un grupo de usuarios ha realizado tras probar la aplicación. Esta evaluación consiste en un formulario cuyos resultados serán presentados también en el capítulo, y finalmente, se llevará a cabo un análisis de estos resultados obtenidos.

5.1 Metodología de la evaluación

Una vez se ha desarrollado el sistema del Proyecto Final de Carrera se ha realizado una evaluación del mismo mediante la ejecución de la aplicación de 10 usuarios (7 hombres y 3 mujeres) con edades comprendidas entre los 25 y los 60 años. Tras la prueba del sistema, se ha llevado a cabo una entrevista ayudada de un cuestionario con el que recoger las opiniones subjetivas de estos usuarios. De este modo, no solo se obtendrá la evaluación del sistema, sino que, además, tenemos la posibilidad de localizar errores.

Para obtener una evaluación lo más completa posible se han analizado los siguientes factores: la experiencia previa del usuario con este tipo de sistemas, conocimiento de la administración de fincas, comprensión por parte del sistema de sus elocuciones, entendimiento de los mensajes reproducidos por el sistema, fluidez en la obtención de la información requerida, dificultad encontrada y satisfacción final.

El cuestionario elaborado para este fin, consta de 10 preguntas y se muestra en la Figura 66.

1.- ¿Cuántas veces ha interactuado oralmente con máquinas?
- Nunca
- Pocas veces
- Con frecuencia

2.- ¿Cuál era su conocimiento de los sistemas de administración de fincas?
- Muy poco
- Poco
- Regular
- Elevado
- Muy elevado

3.- ¿Cómo le ha entendido el sistema lo que quería realizar?
- Muy mal
- Mal
- Regular
- Bien
- Muy bien

4.- ¿Cómo ha sido la claridad de los mensajes recibidos por el sistema?
- Muy mal
- Mal
- Regular
- Bien
- Muy bien

5.- ¿Considera que ha resultado sencillo obtener las indicaciones del sistema?
- Muy sencillo
- Sencillo
- Normal
- Difícil
- Muy difícil

6.- ¿Cómo considera que ha sido la velocidad al obtener las indicaciones del sistema?
- Muy lento
- Lento
- Regular
- Rápido
- Muy rápido

7.- ¿Cree que el sistema se comportó de manera similar a como lo haría un humano?
- Si
- No

8.- En términos generales, ¿está usted satisfecho con el sistema?
- Si
- No

9.- ¿Ha tenido algún problema con el sistema?
- Si
- No

10.- En caso de haber tenido problemas, por favor describa brevemente cuales han sido.

Figura 66. Cuestionario de evaluación

5.2 Resultados de la evaluación

En la siguiente tabla se muestran los resultados de la valoración del sistema según la opinión de los usuarios.

PREGUNTA	RESPUESTA	CANTIDAD
1.- ¿Cuántas veces ha interactuado oralmente con máquinas	Nunca	0
	Pocas veces	5
	Con frecuencia	5
2.- ¿Cuál era su conocimiento de los sistemas de administración de fincas?	Muy poco	1
	Poco	2
	Regular	3
	Elevado	3
	Muy elevado	1
3.- ¿Cómo ha entendido el sistema lo que quería realizar?	Muy mal	0
	Mal	1
	Regular	2
	Bien	7
	Muy bien	1
4.- ¿Cómo ha sido la claridad de los mensajes recibidos por el sistema?	Muy mal	0
	Mal	0
	Regular	0
	Bien	5
	Muy bien	5
5.- ¿Considera que ha resultado sencillo obtener la información del sistema?	Muy sencillo	4
	Sencillo	5
	Normal	1
	Dificil	0
	Muy dificl	0
6.- ¿Cómo considera que ha sido la velocidad al obtener las indicaciones del sistema?	Muy lento	0
	Lento	0
	Regular	2
	Rápido	6
	Muy rápido	2
7.- ¿Cree que el sistema se comportó de manera similar a como lo haría un humano?	Si	9
	No	1
8.- ¿está usted satisfecho con el sistema?	Si	10
	No	0
9.- ¿Ha tenido algún problema con el sistema?	Si	3
	No	7
10.-Problemas encontrados.	- Aunque te ofrece repetir la petición, el sistema no comprende lo que dices demasiadas veces (x2) - A la hora de meter la fecha por el teclado, no sabía cómo hacerlo.	

Tabla 17: Resultados de la evaluación

Analizando los resultados obtenidos obtenemos las siguientes conclusiones:

Actualmente existen muchos sistemas de reconocimiento del habla, por lo tanto, los usuarios ya están bastante habituados al uso de esta tecnología. Para ninguno de los usuarios encuestados era la primera vez que trataban con algo así.

Por el contrario, los sistemas de administración de fincas eran desconocidos para algunos de los usuarios. Esto probablemente es debido a que varios de los encuestados no poseen un piso propio y no están al tanto de este tipo de información, a pesar de este desconocimiento inicial, en líneas generales la información ofrecida por la aplicación resultó de interés y utilidad.

Respecto al sistema, todos o la gran mayoría están satisfechos con la velocidad, con la facilidad a la hora de recibir información y con la fácil comprensión de la información recibida, sin embargo, hay bastante decepción con la comprensión de la información transmitida y más de un usuario transmitió alguna queja sobre ello. A pesar de no entender esta información, el sistema siempre permite volver a repetir la información si no se ha entendido, por lo que finalmente los usuarios pudieron obtener la información deseada.

El sistema guía en todo momento al usuario informando de las distintas opciones que se pueden realizar, por lo que ha resultado sencillo de usar y la navegación ha sido muy intuitiva.

Finalmente, el sistema resultó en líneas generales satisfactorio para todos los usuarios y la mayoría, a pesar de las reticencias conocidas para hablar con máquinas, aceptaron que el programa se comportaba de forma similar a como podría hacerlo un humano al poder entablar un diálogo entre el usuario y la máquina siempre y cuando se ciñeran a las opciones disponibles en el sistema.

Tras esta evaluación el resultado ha sido muy positivo puesto que los usuarios pudieron trabajar con el sistema sin grandes problemas y las opiniones fueron muy favorables.

Capítulo 6

Conclusiones y futuros proyectos

En este capítulo se detallan las conclusiones extraídas durante el desarrollo de todas las fases de este Proyecto Fin de Carrera.

También, en este capítulo se aportarán líneas o trabajos futuros sobre este proyecto con relación a las tecnologías empleadas y funcionalidades ofrecidas.

6.1 Conclusiones

En este Proyecto Final de Carrera se ha desarrollado un sistema basado en el estándar VoiceXML, siendo su principal objetivo realizar una aplicación en este lenguaje de programación.

El sistema desarrollado consiste en una nueva funcionalidad para los programas de administración de fincas existentes en el mercado. Esta funcionalidad permite la consulta de la información básica de una comunidad rápidamente a través del teléfono.

Se ha separado la aplicación en distintos módulos considerando las funcionalidades y las decisiones que el usuario vaya tomando durante el diálogo. Estos elementos se relacionan de tal forma que en una llamada el usuario no esté limitado a una sola acción, pudiendo realizar, dentro de las posibilidades, tantas como considere oportunas.

Para fundamentar la aplicación dentro del campo de los sistemas de diálogo, se ha procedido a la realización de un estudio completo de estos sistemas, que ha quedado extensamente plasmado en el apartado relativo al estado del arte, en el que se describen sus capacidades, los módulos que los conforman y sus principales aplicaciones.

Para desarrollar la aplicación basándose en el potencial de los sistemas de diálogo se ha usado el lenguaje de programación VoiceXML. Este lenguaje facilita mucho el desarrollo del sistema de administración de fincas, permitiendo construir diálogos de forma sencilla y simplificando el reconocimiento de voz mediante las gramáticas, la síntesis de texto a voz o el control de flujo del diálogo.

La actual proliferación y perspectivas de incremento de los sistemas de diálogo remarcan la importancia de este lenguaje. También cabe destacar que cuenta con el apoyo de muchas empresas, por lo que se puede predecir que VoiceXML seguirá creciendo y que la elección de lenguaje utilizado puede considerarse completamente acertada.

La plataforma sobre la que se han implementado todos los desarrollos pertinentes a VoiceXML ha sido Voxeo Evolution. Esta plataforma nos ha proporcionado la infraestructura y los componentes de reconocimiento y síntesis de voz necesarios para nuestro proyecto.

Cabe destacar su sencillez de uso, así como la posibilidad de crear un número al que llamar a las aplicaciones desarrolladas mediante Skype. No debemos olvidar que, además, también hace las veces de servidor, almacenando los ficheros de la aplicación y que tiene un depurador y un foro en el que profesionales responden a las dudas puedan ir surgiendo a medida que se desarrolla y se extiende el uso de la aplicación.

Siguiendo con los aspectos tecnológicos se refiere, la realización de este proyecto no solo ha servido para adquirir conocimientos sobre el estándar VoiceXML, sino que, también ha servido para utilizar y asentar diferentes conceptos adquiridos durante estos años de estudio como son el uso de lenguajes de programación (PHP, SQL) y el uso de bases de datos relacionales (MySQL).

Como principal objetivo del PFC se definía el estudio y desarrollo de un sistema de diálogo usando el lenguaje de programación VoiceXML, procediendo al estudio de la cohesión que este lenguaje tiene con otras tecnologías. Este objetivo se ha cumplido satisfactoriamente, puesto que se ha realizado un estudio intensivo de estas tecnologías y ha sido desarrollada convenientemente la cohesión con la base de datos de un sistema de administración de fincas, pudiendo concluir que el resultado del proyecto queda plasmado en la presente memoria y, de manera fundamental, en la aplicación desarrollada.

Por último, se concluye también que se han cumplido los objetivos secundarios remarcados en el apartado correspondiente, habiéndose satisfecho las metas que recordamos a continuación:

- Accesibilidad: Al ser una aplicación telefónica, sólo se necesita un teléfono para poder acceder a ella.

- Eliminar barreras de acceso: Al ser una aplicación accesible exclusivamente por voz, hemos conseguido que un usuario con discapacidades visuales o motoras pueda acceder al sistema y realizar las consultas necesarias rápidamente y sin ayuda.

- Simplicidad: Se ha conseguido acceder a información que se encuentra en la red a través del teléfono, diseñando una aplicación con una interacción sencilla, donde es fácil navegar y obtener la información que se necesita.

6.2 Trabajos futuros

Tras la finalización de este proyecto, se detallan a continuación los puntos adicionales sobre los que se podría trabajar para obtener mayores prestaciones en la aplicación desarrollada.

APLICACIÓN EN UN SISTEMA REAL

El sistema se ha desarrollado sobre una abstracción de una base de datos similar al de un programa de administración de fincas existente en el mercado. Este sistema debería poder integrarse por completo en un sistema real con poco trabajo.

La integración en un sistema real se realizaría recogiendo la información existente del sistema que se va a integrar y llevándola a nuestra base de datos para trabajar a partir de entonces desde nuestro programa original con el sistema ya integrado.

Otra opción menos deseable sería analizar la base de datos del programa de destino y modificar las consultas SQL de la clase *GestionGramatica.php* para que puedan funcionar en la base de datos del programa original.

AMPLIACIÓN DE LAS OPCIONES Y BASE DE DATOS

La base de datos empleada ha sido simplificada para recoger únicamente la información necesaria para las opciones que ofrece nuestra aplicación. Esta base de datos puede ampliarse tanto con más tablas que recojan nuevas funcionalidades, como con más atributos en las tablas existentes para guardar más información de los elementos existentes.

Del mismo modo que puede ampliarse la base de datos, también pueden ampliarse los módulos de nuestro sistema para poder realizar más consultas que las desarrolladas para este PFC. El sistema puede ampliarse ofreciendo más opciones de contabilidad, un nuevo módulo de reserva y consulta de espacios comunes, consulta de propiedades en venta y/o alquiler, etc…

NUEVOS IDIOMAS

El idioma empleado para la realización de este proyecto ha sido el castellano. Dada la internacionalización tecnológica y la existencia de una aldea global en términos de innovación, una mejora posible para este trabajo vendría representada por una posible traducción del mismo a otros idiomas.

Con la inclusión de nuevos idiomas, no solo podrían utilizar el sistema los usuarios que no hablen castellano, si no que podría exportarse la funcionalidad a programas de administración de fincas de países de habla no hispana.

PERFILES DE NAVEGACIÓN

Si se identificara la necesidad del usuario a partir de sus interacciones anteriores con el sistema, se podría optimizar la navegación por el portal de voz para acortar lo máximo posible el acceso a los servicios más visitados.

Adicionalmente, teniendo en cuenta el número de accesos previos, podrían modificarse los mensajes proporcionados por el sistema. Por ejemplo, si se trata de un usuario que lo utilizara por primera vez, se darían completas indicaciones del funcionamiento para facilitar el uso y disminuir los errores que se pudieran producir. En cambio, si el usuario ya fuera experto en su utilización, no habría que explicar qué decir exactamente en cada momento y se ofrecerían directamente las operaciones que en su historial resultaran más comunes, consiguiendo de esta forma una mayor velocidad en realizar la acción deseada.

Capítulo 7

Presupuesto

En esta sección se presenta el presupuesto del proyecto. En él se contempla la duración de las distintas fases y tareas, y se incluye un desglose de costes de personal, costes de material y costes totales.

TAREAS

Fase de planificación:

- Estudio de los sistemas de diálogo.
  Duración: 20 horas.

- Estudio de los sistemas de administración de fincas.
  Duración: 10 horas.

- Estudio y práctica del lenguaje VoiceXML.
  Duración: 80 horas.

- Estudio y preparación del servidor web.
  Duración: 10 horas.

- Estudio de las tecnologías y lenguajes necesarios.
  Duración: 20 horas.

Fase de desarrollo:

- Diseño de la base de datos.
  Duración: 40 horas.

- Diseño del flujo de interacción.
  Duracion: 40 horas.

- Diseño de gramáticas.
  Duracion: 60 horas.

- Desarrollo de la aplicación.
  Duracion: 240 horas.

- Pruebas.
  Duracion: 120 horas.

- Evaluación.
  Duracion: 20 horas.

Fase de documentación:

- Memoria del Proyecto Final de Carrera.
  Duracion: 100 horas.

- Preparación de la presentación.
  Duracion: 20 horas.

RECURSOS

Los medios utilizados para llevar a cabo el Proyecto Final de Carrera son los siguientes:

- **Recursos software:**
  - Mozilla Firefox. 0€
  - Plataforma Voxeo. 0€
  - Skype. 0€
  - Microsoft Office 2016. 190€
  - EditPlus Text Editor. 0€
- **Recursos hardware**

- Ordenador portátil. 800€
- Periféricos habituales (teclado, ratón). 10€
- Impresora HP Deskjet 3637. 50€
- Auriculares con micrófono. 25€
- Servidor 000webhost. 0€
- Router. 0€

- **Recursos humanos**

En la realización de este proyecto han participado dos personas, el director de proyecto y el desarrollador.

- Coste de un ingeniero. 35€ / hora

RESUMEN DE COSTES

CONCEPTO	IMPORTE
TAREAS: 800 horas	28.000 €
RECURSOS	1.075 €
SUBTOTAL	29.075 €
IVA (21%)	6.105,75 €
TOTAL	**35.180,75 €**

Tabla 18: Resumen de costes

El presupuesto total de este proyecto asciende a la cantidad de **TREINTA Y CINCO MIL CIENTO OCHENTA EUROS CON SETENTA Y CINCO CÉNTIMOS.**

Leganés a 04 de septiembre de 2017

Fdo. Felipe Carlos Ballesteros Costero

Glosario

CCXML	*Call Control Extensible Language*
CD	*Compact Disc*
DMTF	*Dual Tone Multiple Frequency*
DVD	*Digital Versatile Disc*
ECMA	*European Computer Manufacturers Association*
FIA	*Form Interpretation Algorithm*
GB	*Gigabyte*
HP	*Hewlett-Packard*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IBM	*International Business Machines*
IVR	*Interactive Voice Response*
MySQL	*My Structured Query Language*
PDA	*Personal Digital Assistant*
PHP	*Hypertext Preprocessor*
PML	*Phone Markup Language*
RTC	*Red Telefónica Conmutada*
SRGS	*Speech Recognition Grammar Specification*
SSML	*Speech Synthesis Markup Language.*
SQL	*Structured Query Language*
TIC	*Tecnologías de Información y Comunicación*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
VoiceXML	*Voice eXtensible Markup Language*
VoIP	*Voice Over Internet Protocol*
W3C	*World Wide Web Consortium*
WAP	*Wireless Application Protocol*
XHTML	*eXtensible Hypertext Markup Language*
XML	*eXtensible Markup Language*

Referencias

[ALEXA] *Alexa, el asistente virtual de Amazon, llega al Iphone*. Noticia de La Vanguardia. Marzo2017. http://www.lavanguardia.com/tecnologia/20170317/42954704667/amazon-alexa-iphone-siri-asistentes-virtuales-cortana-google.html

[ALV06] Álvarez García, Victor. *Estándar VoiceXML*. Disponible: http://di002.edv.uniovi.es/~cueva/asignaturas/doctorado/2006/trabajos/VoiceXML.pdf

[BOE60] *Ley 49/1960, de 21 de julio, sobre propiedad horizontal*. Disponible: https://boe.es/buscar/pdf/1960/BOE-A-1960-10906-consolidado.pdf

[BOHUS] A list of spoken languaje interfaces http://www.cs.cmu.edu/~dbohus/SDS/

[BUSLINE] Aplicación para encontrar autobuses en Oakland. http://www.cs.cmu.edu/~aria/BusLine.html

[CCXML] Voice Browser Call Control: CCXML Version 1.0. Julio 2011. https://www.w3.org/TR/ccxml/

[COZAR] López-Cózar, Ramón – Granell, Ramón. *Sistema de Diálogo Basado en VoiceXML para Proporcionar Información de Viajes en Tren.* http://www.sepln.org/revistaSEPLN/revista/33/33-Pag171.pdf

[DTMF] DTMF Grammars. https://msdn.microsoft.com/en-us/library/ee800147.aspx

[FINCASPRO] Empresa FincasPro. http://www.fincaspro.es/

[GRI07] Griol, David. *Desarrollo y evaluación de Diferentes Metodologías para la Gestión Automática del Diálogo*. Tesis Doctoral. Universidad Politécnica de Valencia. Noviembre 2007. Disponible: http://www.dsic.upv.es/docs/bib-dig/tesis/etd-07272007-102625/tesisdgriol.pdf

[IRENE] Asistente virtual de Renfe. http://consulta.renfe.com/base/main

[JUPITER] Sistema de conversación que ofrece información metereológica. http://groups.csail.mit.edu/sls/research/jupiter.shtml

[LARS06] Dialogue Systems and Projects. http://www.ling.gu.se/~sl/dialogue_links.html

[LASO] Empresa Laso S.L. http://www.lasosl.com/

[LETSGO] Sistema de diálogo oral que proporciona acceso a la ruta de autobuses e información de la programación en Pittsburgh. http://www.speech.cs.cmu.edu/letsgo/

[LISTEN] Tutor de lectura automatizado. http://www.cs.cmu.edu/~listen/

[LLIS03] Llisterri, Joaquim – Carbó, Carme – Machuca, María Jesús, de la Mota, Carme – Riera, Montserrat – Ríos, Antonio. Universitat Autònoma de Barcelona. *El papel de la lingüística en el desarrollo de las tecnologías del habla*. Disponible: https://www.researchgate.net/publication/242085674_El_papel_de_la_linguistica_en_el_desarrollo_de_las_tecnologias_del_habla

[LLI06] Llisterri, J. - Machuca, M. J. *Los sistemas de diálogo*. Soria: Universitat Autònoma de Barcelona, Servei de Publicacions - Fundación Duques de Soria, 2006. Disponible: http://liceu.uab.es/~joaquim/speech_technology/tecnol_parla/dialogue/dialogue_multimodal/multimodalidad_dialogo.html#Los_sistemas_de_di_logo_multimodales

[LOP06] López-Cózar Delgado, Ramón – Araki, Masahiro: Spoken, Multilingual and Multimodal Dialogue Systems: Development and Assessment. 2005

[LOP07] Curso: "Procesamiento del Habla e Interacción Multimodal". 2007. Disponible: http://www.ugr.es/~rlopezc/phim.htm

[MERCURY] Sistema que proporciona información de vuelos. http://groups.csail.mit.edu/sls/research/mercury.shtml

[NATURALVOX] Empresa Natural Vox. http://www.naturalvox.eu/es/inicio/

[NETFINCAS] Empresa NetFincas. http://www.netfincasweb.com/

[ORACLE] Oracle MySQL. https://www.oracle.com/es/mysql/index.html

[PEGASUS] Sistema de conversación que proporciona información del estado de un vuelo. http://groups.csail.mit.edu/sls/research/pegasus.shtml

[PHP] Manual online de PHP. http://php.net/

[PORTALFINCAS] PortalFincas. http://www.portalfincas.com/

[SIRI] Asistente virtual de Apple. https://www.ipadizate.es/tag/siri/

[SRGS] Speech Recognition Grammar Specification Version 1.0. Marzo 2004. https://www.w3.org/TR/speech-grammar/

[SSML] Speech Synthesis Markup Languaje Version 1.0. Septiembre 2004. https://www.w3.org/TR/speech-synthesis/

[TELLME] Ayuda a desarrolladores de VoiceXML. https://studio.tellme.com/

[USI] Universal Speech Interface. http://www.cs.cmu.edu/~usi/

[VERBIO] VoiceXML Extensible Markup Languaje. Guía del Usuario. Versión 7.1. Enero 2005. http://www.verbio.com/webverbiotm/html/reference/pdf/guide_voicexml_es.pdf

[VOXEO] Voxeo's Global Developer and Support Portal. https://evolution.voxeo.com/

[VOYAGER] Sistema de información turística. http://groups.csail.mit.edu/sls/research/web.shtml

[VoiceXML] *Extensible Markup Language Version 3.0* - W3C Working Draft 16 December 2010. http://www.w3.org/TR/voicexml30/

[WAXHOLM] Sistema de información sobre el tráfico de barcos. http://www.speech.kth.se/waxholm/waxholm2.html

[WEBHOST] Servidor web 000webhost. https://www.000webhost.com/

[XML] *Extensible Markup Language (XML) 1.0 (Fifth Edition)* – W3C Recommendation 26 November 2008. https://www.w3.org/TR/xml/

[YUKEI] *Trabajar con fechas en MySQL*. Yukey.net. Octubre 2014. https://www.yukei.net/2014/10/trabajar-con-fechas-en-mysql/